

EXHIBIT 1

BUSINESS

5 things to know about Amazon's new mega-warehouse on Boxwood Road in Delaware

**Brandon Holveck**

Delaware News Journal

Published 5:28 p.m. ET Sept. 21, 2021 | Updated 9:09 p.m. ET Sept. 25, 2021

When Amazon's mega-warehouse on Boxwood Road near Newport stowed its first product last week – a plush dog toy – it became the largest operational Amazon facility in the country.

Covering roughly 640,000 square feet of ground space and five stories high, the new warehouse totals 3.8 million square feet, making it more than three times the size of Amazon's Middletown fulfillment center.

"The size is huge," said Amazon manager Qadir Stevenson.

But it's not just the size that sets the Boxwood Road facility apart. The fulfillment center features all of Amazon's latest technology, including robots that move products miles every day and sensors that detect when and where a product is stowed.

Amazon invited media on a tour of the facility Tuesday afternoon. Here's what we saw.

PREVIOUS REPORTING: Amazon's largest warehouse opens soon in Delaware. Up to 1,000 workers, robots to run high-tech site.

1. How does it work?

Nothing is manufactured at the Boxwood Road facility.

Amazon sellers send products to the warehouse where they are stored and eventually packaged for delivery to customers. The game is getting products from one point to another as efficiently as possible.

The first station products are sent to once they arrive at the warehouse is "decont," or decontainer. There, employees put products into yellow totes, which are then sent to different points across the five-story building via a winding series of conveyor belts.

The totes arrive at a stowing station where an employee puts each item into a "robotics pod." The pod looks like a large shelving unit with about a dozen small compartments on each face.

The products stowed in the robotics pods range anywhere from makeup to over-the-counter medicine and LED stripe lights. It's at this point the products appear as available on Amazon.com.

At similar looking stations, employees pick items from the pods to fulfill online orders and place them in totes that are sent elsewhere to be packaged and delivered.

The company has a device called an AFE or Amazon fulfillment engine that allows multiple products to be packaged in the same box. Amazon officials did not explain the process, citing proprietary reasons.

The Boxwood Road facility sends packages to sortation centers, which sort the packages for delivery by zip code. Packages are then sent to delivery stations, which handle the "last mile" of the process to the customer's front door.

At the back of the Boxwood Road property Amazon built a second smaller building that serves as a delivery station.

2. Robots? What do the robots do?

The upper four floors are where the robots live. They look like oversized Roomba vacuum cleaners and transport roughly 40,000 robotics pods across the facility (10,000 fit on a floor).

The robots move along a series of "alleys and roads" to the 128 stations on each floor manned by human employees. The robots are sent back and forth to the employee stations by a computer system.

At the stations, employees are bound by a wall of totes and a computer screen to one side and the robotics pod – that shelving unit of various products – to the other. Each station is dedicated to one purpose. They either store products, pick products from storage for delivery or count products to maintain an accurate inventory.

When they aren't in use, the robotics pods are returned to their spots in a perfectly formed grid covering thousands of square feet of dimly lit floor space. The robotics pods are surrounded by a chain-link fence.

Amazon touts its robotics system as a win for employee safety. It also makes the operation more efficient. Employees don't spend their day walking through shelves as though they're stocking or shopping at a massive store. The would-be shelving units can hold items on all four sides, not just the one facing outward.

DELAWARE WAREHOUSES: Online shopping, highways and nearby big cities leading to a boom in warehouses in Delaware

3. Other technology and observations

The stowing stations at the facility have a vertical light filter that detects when an employee reaches their hand into the robotics pod where products are stored.

In tandem with the computer system, which directs where products are moving from and to, the system allows stowers to sort products without having to scan each one before moving them. The containers also sense the weight of the product. The light filter was not shown to media.

Most of the interior of the facility was gray with yellow accenting. With no windows, it was all subject to the white glow of the overhead LED light panels. The parts of the warehouse shown to media were quiet.

4. Working at Amazon

Amazon has hired about 500 people for the Boxwood Road facility. It expects to hire about 500 more workers before the end of the year. Officials are also aiming for the facility to be fully operational at that time.

The stowers and pickers whose work was observed by reporters work 10- or 12-hour shifts, said General Manager Will Carney.

The minimum wage for the site on a per-hour basis is \$16.25, and employees can receive a sign-on bonus of up to \$3,000. Other benefits include a 401(k) savings plan and a college tuition program.

5. Amazon has history with Delaware

Amazon opened its first fulfillment center some 24 years ago in Delaware.

Still active today, the fulfillment center off Frenchtown Road near New Castle is smaller than the size of one floor of the new Boxwood Road facility.

Amazon has at least eight facilities planned or opened in the First State, according to media reports. They include plans for a new fulfillment center at the former Blue Diamond Park on Hamburg Road near New Castle and delivery stations in Delaware City and Seaford.

Like knowing what stores, restaurants and developments are coming and going in Delaware? Join our Facebook group [What's Going There in Delaware](#) and subscribe to our [What's Going There in Delaware](#) newsletter.

Contact Brandon Holveck at bholveck@delawareonline.com. Follow him on Twitter [@holveck_brandon](#).

EXHIBIT 2

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.264

(03/2005)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

Infrastructure of audiovisual services – Coding of moving
video

Advanced video coding for generic audiovisual services

ITU-T Recommendation H.264

ITU-T H-SERIES RECOMMENDATIONS
AUDIOVISUAL AND MULTIMEDIA SYSTEMS

CHARACTERISTICS OF VISUAL TELEPHONE SYSTEMS	H.100–H.199
INFRASTRUCTURE OF AUDIOVISUAL SERVICES	
General	H.200–H.219
Transmission multiplexing and synchronization	H.220–H.229
Systems aspects	H.230–H.239
Communication procedures	H.240–H.259
Coding of moving video	H.260–H.279
Related systems aspects	H.280–H.299
Systems and terminal equipment for audiovisual services	H.300–H.349
Directory services architecture for audiovisual and multimedia services	H.350–H.359
Quality of service architecture for audiovisual and multimedia services	H.360–H.369
Supplementary services for multimedia	H.450–H.499
MOBILITY AND COLLABORATION PROCEDURES	
Overview of Mobility and Collaboration, definitions, protocols and procedures	H.500–H.509
Mobility for H-Series multimedia systems and services	H.510–H.519
Mobile multimedia collaboration applications and services	H.520–H.529
Security for mobile multimedia systems and services	H.530–H.539
Security for mobile multimedia collaboration applications and services	H.540–H.549
Mobility interworking procedures	H.550–H.559
Mobile multimedia collaboration inter-working procedures	H.560–H.569
BROADBAND AND TRIPLE-PLAY MULTIMEDIA SERVICES	
Broadband multimedia services over VDSL	H.610–H.619

For further details, please refer to the list of ITU-T Recommendations.

ITU-T Recommendation H.264

Advanced video coding for generic audiovisual services

Summary

This Recommendation | International Standard represents an evolution of the existing video coding standards (H.261, H.262, and H.263) and it was developed in response to the growing need for higher compression of moving pictures for various applications such as videoconferencing, digital storage media, television broadcasting, Internet streaming, and communication. It is also designed to enable the use of the coded video representation in a flexible manner for a wide variety of network environments. The use of this Recommendation | International Standard allows motion video to be manipulated as a form of computer data and to be stored on various storage media, transmitted and received over existing and future networks and distributed on existing and future broadcasting channels.

The revision approved 2005-03 contains modifications of the video coding standard to add four new profiles, referred to as the High, High 10, High 4:2:2, and High 4:4:4 profiles, to improve video quality capability and to extend the range of applications addressed by the standard (for example, by including support for a greater range of picture sample precision and higher-resolution chroma formats). Additionally, a definition of new types of supplemental data has been specified to further broaden the applicability of the video coding standard. Finally, a number of corrections to errors in the published text have been included. This revision, in addition to enhancing video coding capability, will serve to maintain technical alignment with the corresponding jointly-developed ISO/IEC 14496-10 standard.

Corrigendum 1 to ITU-T Rec. H.264 corrected and updated various minor aspects to bring the ITU-T version of the text up to date relative to the April 2005 output status approved as a new edition of the corresponding jointly-developed and technically-aligned text ISO/IEC 14496-10. It additionally fixes a number of minor errors and needs for clarification and defines three previously-reserved sample aspect ratio indicators.

This edition includes the text approved 2005-03 and its Corrigendum 1 approved 2005-09.

Source

ITU-T Recommendation H.264 was approved on 1 March 2005 by ITU-T Study Group 16 (2005-2008) under the ITU-T Recommendation A.8 procedure. It includes modifications introduced by H.264 (2005) Cor.1 approved on 13 September 2005 by ITU-T Study Group 16 (2005-2008) under the ITU-T Recommendation A.8 procedure.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure e.g. interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had received notice of intellectual property, protected by patents, which may be required to implement this Recommendation. However, implementors are cautioned that this may not represent the latest information and are therefore strongly urged to consult the TSB patent database.

© ITU 2005

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

CONTENTS

	Page
Foreword	xiv
0 Introduction	1
0.1 Prologue	1
0.2 Purpose	1
0.3 Applications	1
0.4 Publication and versions of this specification	1
0.5 Profiles and levels	2
0.6 Overview of the design characteristics	2
0.6.1 Predictive coding	3
0.6.2 Coding of progressive and interlaced video	3
0.6.3 Picture partitioning into macroblocks and smaller partitions	3
0.6.4 Spatial redundancy reduction	3
0.7 How to read this specification	3
1 Scope	4
2 Normative references	4
3 Definitions	4
4 Abbreviations	12
5 Conventions	12
5.1 Arithmetic operators	13
5.2 Logical operators	13
5.3 Relational operators	13
5.4 Bit-wise operators	13
5.5 Assignment operators	14
5.6 Range notation	14
5.7 Mathematical functions	14
5.8 Variables, syntax elements, and tables	15
5.9 Text description of logical operations	16
5.10 Processes	17
6 Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships	17
6.1 Bitstream formats	17
6.2 Source, decoded, and output picture formats	17
6.3 Spatial subdivision of pictures and slices	22
6.4 Inverse scanning processes and derivation processes for neighbours	23
6.4.1 Inverse macroblock scanning process	23
6.4.2 Inverse macroblock partition and sub-macroblock partition scanning process	24
6.4.2.1 Inverse macroblock partition scanning process	25
6.4.2.2 Inverse sub-macroblock partition scanning process	25
6.4.3 Inverse 4x4 luma block scanning process	26
6.4.4 Inverse 8x8 luma block scanning process	26
6.4.5 Derivation process of the availability for macroblock addresses	26
6.4.6 Derivation process for neighbouring macroblock addresses and their availability	27
6.4.7 Derivation process for neighbouring macroblock addresses and their availability in MBAFF frames	27
6.4.8 Derivation process for neighbouring macroblocks, blocks, and partitions	28
6.4.8.1 Derivation process for neighbouring macroblocks	29
6.4.8.2 Derivation process for neighbouring 8x8 luma block	29
6.4.8.3 Derivation process for neighbouring 4x4 luma blocks	30
6.4.8.4 Derivation process for neighbouring 4x4 chroma blocks	30
6.4.8.5 Derivation process for neighbouring partitions	31
6.4.9 Derivation process for neighbouring locations	33
6.4.9.1 Specification for neighbouring locations in fields and non-MBAFF frames	33
6.4.9.2 Specification for neighbouring locations in MBAFF frames	34
7 Syntax and semantics	36
7.1 Method of describing syntax in tabular form	36
7.2 Specification of syntax functions, categories, and descriptors	37
7.3 Syntax in tabular form	38
7.3.1 NAL unit syntax	38
7.3.2 Raw byte sequence payloads and RBSP trailing bits syntax	39
7.3.2.1 Sequence parameter set RBSP syntax	39

7.3.2.1.1	Scaling list syntax	40
7.3.2.1.2	Sequence parameter set extension RBSP syntax	40
7.3.2.2	Picture parameter set RBSP syntax	41
7.3.2.3	Supplemental enhancement information RBSP syntax	42
7.3.2.3.1	Supplemental enhancement information message syntax	42
7.3.2.4	Access unit delimiter RBSP syntax	43
7.3.2.5	End of sequence RBSP syntax	43
7.3.2.6	End of stream RBSP syntax	43
7.3.2.7	Filler data RBSP syntax	43
7.3.2.8	Slice layer without partitioning RBSP syntax	43
7.3.2.9	Slice data partition RBSP syntax	43
7.3.2.9.1	Slice data partition A RBSP syntax	43
7.3.2.9.2	Slice data partition B RBSP syntax	44
7.3.2.9.3	Slice data partition C RBSP syntax	44
7.3.2.10	RBSP slice trailing bits syntax	44
7.3.2.11	RBSP trailing bits syntax	44
7.3.3	Slice header syntax	45
7.3.3.1	Reference picture list reordering syntax	46
7.3.3.2	Prediction weight table syntax	47
7.3.3.3	Decoded reference picture marking syntax	48
7.3.4	Slice data syntax	49
7.3.5	Macroblock layer syntax	50
7.3.5.1	Macroblock prediction syntax	51
7.3.5.2	Sub-macroblock prediction syntax	52
7.3.5.3	Residual data syntax	53
7.3.5.3.1	Residual block CAVLC syntax	54
7.3.5.3.2	Residual block CABAC syntax	55
7.4	<i>Semantics</i>	56
7.4.1	NAL unit semantics	56
7.4.1.1	Encapsulation of an SODB within an RBSP (informative)	58
7.4.1.2	Order of NAL units and association to coded pictures, access units, and video sequences	59
7.4.1.2.1	Order of sequence and picture parameter set RBSPs and their activation	59
7.4.1.2.2	Order of access units and association to coded video sequences	60
7.4.1.2.3	Order of NAL units and coded pictures and association to access units	60
7.4.1.2.4	Detection of the first VCL NAL unit of a primary coded picture	62
7.4.1.2.5	Order of VCL NAL units and association to coded pictures	63
7.4.2	Raw byte sequence payloads and RBSP trailing bits semantics	63
7.4.2.1	Sequence parameter set RBSP semantics	63
7.4.2.1.1	Scaling list semantics	68
7.4.2.1.2	Sequence parameter set extension RBSP semantics	69
7.4.2.2	Picture parameter set RBSP semantics	70
7.4.2.3	Supplemental enhancement information RBSP semantics	73
7.4.2.3.1	Supplemental enhancement information message semantics	73
7.4.2.4	Access unit delimiter RBSP semantics	73
7.4.2.5	End of sequence RBSP semantics	73
7.4.2.6	End of stream RBSP semantics	73
7.4.2.7	Filler data RBSP semantics	73
7.4.2.8	Slice layer without partitioning RBSP semantics	73
7.4.2.9	Slice data partition RBSP semantics	74
7.4.2.9.1	Slice data partition A RBSP semantics	74
7.4.2.9.2	Slice data partition B RBSP semantics	74
7.4.2.9.3	Slice data partition C RBSP semantics	74
7.4.2.10	RBSP slice trailing bits semantics	74
7.4.2.11	RBSP trailing bits semantics	75
7.4.3	Slice header semantics	75
7.4.3.1	Reference picture list reordering semantics	80
7.4.3.2	Prediction weight table semantics	81
7.4.3.3	Decoded reference picture marking semantics	82
7.4.4	Slice data semantics	85
7.4.5	Macroblock layer semantics	85
7.4.5.1	Macroblock prediction semantics	92
7.4.5.2	Sub-macroblock prediction semantics	93
7.4.5.3	Residual data semantics	95

7.4.5.3.1	Residual block CAVLC semantics.....	96
7.4.5.3.2	Residual block CABAC semantics	96
8	Decoding process.....	96
8.1	NAL unit decoding process.....	97
8.2	Slice decoding process.....	98
8.2.1	Decoding process for picture order count	98
8.2.1.1	Decoding process for picture order count type 0	99
8.2.1.2	Decoding process for picture order count type 1	100
8.2.1.3	Decoding process for picture order count type 2	101
8.2.2	Decoding process for macroblock to slice group map	102
8.2.2.1	Specification for interleaved slice group map type	103
8.2.2.2	Specification for dispersed slice group map type.....	103
8.2.2.3	Specification for foreground with left-over slice group map type	104
8.2.2.4	Specification for box-out slice group map types.....	104
8.2.2.5	Specification for raster scan slice group map types	105
8.2.2.6	Specification for wipe slice group map types	105
8.2.2.7	Specification for explicit slice group map type.....	105
8.2.2.8	Specification for conversion of map unit to slice group map to macroblock to slice group map	105
8.2.3	Decoding process for slice data partitioning	105
8.2.4	Decoding process for reference picture lists construction.....	106
8.2.4.1	Decoding process for picture numbers.....	107
8.2.4.2	Initialisation process for reference picture lists.....	107
8.2.4.2.1	Initialisation process for the reference picture list for P and SP slices in frames	108
8.2.4.2.2	Initialisation process for the reference picture list for P and SP slices in fields	108
8.2.4.2.3	Initialisation process for reference picture lists for B slices in frames	109
8.2.4.2.4	Initialisation process for reference picture lists for B slices in fields	109
8.2.4.2.5	Initialisation process for reference picture lists in fields	110
8.2.4.3	Reordering process for reference picture lists.....	110
8.2.4.3.1	Reordering process of reference picture lists for short-term reference pictures	111
8.2.4.3.2	Reordering process of reference picture lists for long-term reference pictures	112
8.2.5	Decoded reference picture marking process	112
8.2.5.1	Sequence of operations for decoded reference picture marking process.....	113
8.2.5.2	Decoding process for gaps in frame_num.....	113
8.2.5.3	Sliding window decoded reference picture marking process	114
8.2.5.4	Adaptive memory control decoded reference picture marking process	114
8.2.5.4.1	Marking process of a short-term reference picture as “unused for reference”	114
8.2.5.4.2	Marking process of a long-term reference picture as “unused for reference”	115
8.2.5.4.3	Assignment process of a LongTermFrameIdx to a short-term reference picture	115
8.2.5.4.4	Decoding process for MaxLongTermFrameIdx	115
8.2.5.4.5	Marking process of all reference pictures as “unused for reference” and setting MaxLongTermFrameIdx to “no long-term frame indices”	115
8.2.5.4.6	Process for assigning a long-term frame index to the current picture	116
8.3	Intra prediction process.....	116
8.3.1	Intra_4x4 prediction process for luma samples	117
8.3.1.1	Derivation process for the Intra4x4PredMode	117
8.3.1.2	Intra_4x4 sample prediction	119
8.3.1.2.1	Specification of Intra_4x4_Veritical prediction mode	119
8.3.1.2.2	Specification of Intra_4x4_Horizontal prediction mode	119
8.3.1.2.3	Specification of Intra_4x4_DC prediction mode	120
8.3.1.2.4	Specification of Intra_4x4_Diagonal_Down_Left prediction mode	120
8.3.1.2.5	Specification of Intra_4x4_Diagonal_Down_Right prediction mode	120
8.3.1.2.6	Specification of Intra_4x4_Veritical_Right prediction mode	121
8.3.1.2.7	Specification of Intra_4x4_Horizontal_Down prediction mode	121
8.3.1.2.8	Specification of Intra_4x4_Veritical_Left prediction mode.....	122
8.3.1.2.9	Specification of Intra_4x4_Horizontal_Up prediction mode	122
8.3.2	Intra_8x8 prediction process for luma samples	122
8.3.2.1	Derivation process for Intra8x8PredMode.....	123
8.3.2.2	Intra_8x8 sample prediction	124
8.3.2.2.1	Reference sample filtering process for Intra_8x8 sample prediction	125
8.3.2.2.2	Specification of Intra_8x8_Veritical prediction mode	126
8.3.2.2.3	Specification of Intra_8x8_Horizontal prediction mode	126
8.3.2.2.4	Specification of Intra_8x8_DC prediction mode	127
8.3.2.2.5	Specification of Intra_8x8_Diagonal_Down_Left prediction mode	127

8.3.2.2.6	Specification of Intra_8x8_Diagonal_Down_Right prediction mode	127
8.3.2.2.7	Specification of Intra_8x8_Vertical_Right prediction mode	128
8.3.2.2.8	Specification of Intra_8x8_Horizontal_Down prediction mode	128
8.3.2.2.9	Specification of Intra_8x8_Vertical_Left prediction mode	129
8.3.2.2.10	Specification of Intra_8x8_Horizontal_Up prediction mode	129
8.3.3	Intra_16x16 prediction process for luma samples	129
8.3.3.1	Specification of Intra_16x16_Vertical prediction mode	130
8.3.3.2	Specification of Intra_16x16_Horizontal prediction mode	130
8.3.3.3	Specification of Intra_16x16_DC prediction mode	130
8.3.3.4	Specification of Intra_16x16_Plane prediction mode	131
8.3.4	Intra prediction process for chroma samples	131
8.3.4.1	Specification of Intra_Chroma_DC prediction mode	132
8.3.4.2	Specification of Intra_Chroma_Horizontal prediction mode	134
8.3.4.3	Specification of Intra_Chroma_Vertical prediction mode	134
8.3.4.4	Specification of Intra_Chroma_Plane prediction mode	134
8.3.5	Sample construction process for I_PCM macroblocks	134
8.4	<i>Inter prediction process</i>	135
8.4.1	Derivation process for motion vector components and reference indices	137
8.4.1.1	Derivation process for luma motion vectors for skipped macroblocks in P and SP slices	138
8.4.1.2	Derivation process for luma motion vectors for B_Skip, B_Direct_16x16, and B_Direct_8x8	139
8.4.1.2.1	Derivation process for the co-located 4x4 sub-macroblock partitions	139
8.4.1.2.2	Derivation process for spatial direct luma motion vector and reference index prediction mode ...	142
8.4.1.2.3	Derivation process for temporal direct luma motion vector and reference index prediction mode	144
8.4.1.3	Derivation process for luma motion vector prediction	146
8.4.1.3.1	Derivation process for median luma motion vector prediction	147
8.4.1.3.2	Derivation process for motion data of neighbouring partitions	148
8.4.1.4	Derivation process for chroma motion vectors	149
8.4.2	Decoding process for Inter prediction samples	149
8.4.2.1	Reference picture selection process	150
8.4.2.2	Fractional sample interpolation process	151
8.4.2.2.1	Luma sample interpolation process	152
8.4.2.2.2	Chroma sample interpolation process	155
8.4.2.3	Weighted sample prediction process	156
8.4.2.3.1	Default weighted sample prediction process	156
8.4.2.3.2	Weighted sample prediction process	157
8.5	<i>Transform coefficient decoding process and picture construction process prior to deblocking filter process</i>	159
8.5.1	Specification of transform decoding process for 4x4 luma residual blocks	160
8.5.2	Specification of transform decoding process for luma samples of Intra_16x16 macroblock prediction mode	160
8.5.3	Specification of transform decoding process for 8x8 luma residual blocks	161
8.5.4	Specification of transform decoding process for chroma samples	162
8.5.5	Inverse scanning process for transform coefficients	164
8.5.6	Inverse scanning process for 8x8 luma transform coefficients	164
8.5.7	Derivation process for the chroma quantisation parameters and scaling function	166
8.5.8	Scaling and transformation process for luma DC transform coefficients for Intra_16x16 macroblock type	168
8.5.9	Scaling and transformation process for chroma DC transform coefficients	169
8.5.10	Scaling and transformation process for residual 4x4 blocks	171
8.5.11	Scaling and transformation process for residual 8x8 luma blocks	173
8.5.12	Picture construction process prior to deblocking filter process	176
8.5.13	Residual colour transform process	177
8.6	<i>Decoding process for P macroblocks in SP slices or SI macroblocks</i>	178
8.6.1	SP decoding process for non-switching pictures	178
8.6.1.1	Luma transform coefficient decoding process	178
8.6.1.2	Chroma transform coefficient decoding process	179
8.6.2	SP and SI slice decoding process for switching pictures	181
8.6.2.1	Luma transform coefficient decoding process	181
8.6.2.2	Chroma transform coefficient decoding process	181
8.7	<i>Deblocking filter process</i>	182
8.7.1	Filtering process for block edges	186
8.7.2	Filtering process for a set of samples across a horizontal or vertical block edge	188
8.7.2.1	Derivation process for the luma content dependent boundary filtering strength	188
8.7.2.2	Derivation process for the thresholds for each block edge	190
8.7.2.3	Filtering process for edges with bS less than 4	191

8.7.2.4	Filtering process for edges for bS equal to 4.....	193
9	Parsing process.....	194
9.1	<i>Parsing process for Exp-Golomb codes.....</i>	<i>194</i>
9.1.1	Mapping process for signed Exp-Golomb codes	195
9.1.2	Mapping process for coded block pattern	196
9.2	<i>CAVLC parsing process for transform coefficient levels.....</i>	<i>198</i>
9.2.1	Parsing process for total number of transform coefficient levels and trailing ones	199
9.2.2	Parsing process for level information	201
9.2.2.1	Parsing process for level_prefix.....	202
9.2.3	Parsing process for run information.....	203
9.2.4	Combining level and run information	206
9.3	<i>CABAC parsing process for slice data</i>	<i>206</i>
9.3.1	Initialisation process	207
9.3.1.1	Initialisation process for context variables.....	208
9.3.1.2	Initialisation process for the arithmetic decoding engine.....	218
9.3.2	Binarization process.....	219
9.3.2.1	Unary (U) binarization process	221
9.3.2.2	Truncated unary (TU) binarization process	221
9.3.2.3	Concatenated unary/ k-th order Exp-Golomb (UEGk) binarization process	222
9.3.2.4	Fixed-length (FL) binarization process.....	222
9.3.2.5	Binarization process for macroblock type and sub-macroblock type	222
9.3.2.6	Binarization process for coded block pattern.....	225
9.3.2.7	Binarization process for mb_qp_delta	225
9.3.3	Decoding process flow.....	225
9.3.3.1	Derivation process for ctxIdx.....	226
9.3.3.1.1	Assignment process of ctxIdxInc using neighbouring syntax elements	228
9.3.3.1.1.1	Derivation process of ctxIdxInc for the syntax element mb_skip_flag.....	228
9.3.3.1.1.2	Derivation process of ctxIdxInc for the syntax element mb_field_decoding_flag.....	228
9.3.3.1.1.3	Derivation process of ctxIdxInc for the syntax element mb_type.....	229
9.3.3.1.1.4	Derivation process of ctxIdxInc for the syntax element coded_block_pattern.....	229
9.3.3.1.1.5	Derivation process of ctxIdxInc for the syntax element mb_qp_delta	230
9.3.3.1.1.6	Derivation process of ctxIdxInc for the syntax elements ref_idx_l0 and ref_idx_l1.....	230
9.3.3.1.1.7	Derivation process of ctxIdxInc for the syntax elements mvd_l0 and mvd_l1	231
9.3.3.1.1.8	Derivation process of ctxIdxInc for the syntax element intra_chroma_pred_mode.....	232
9.3.3.1.1.9	Derivation process of ctxIdxInc for the syntax element coded_block_flag	233
9.3.3.1.1.10	Derivation process of ctxIdxInc for the syntax element transform_size_8x8_flag	234
9.3.3.1.2	Assignment process of ctxIdxInc using prior decoded bin values	234
9.3.3.1.3	Assignment process of ctxIdxInc for syntax elements significant_coeff_flag, last_significant_coeff_flag, and coeff_abs_level_minus1	235
9.3.3.2	Arithmetic decoding process.....	237
9.3.3.2.1	Arithmetic decoding process for a binary decision	238
9.3.3.2.1.1	State transition process.....	239
9.3.3.2.2	Renormalization process in the arithmetic decoding engine	241
9.3.3.2.3	Bypass decoding process for binary decisions	242
9.3.3.2.4	Decoding process for binary decisions before termination	242
9.3.4	Arithmetic encoding process (informative)	243
9.3.4.1	Initialisation process for the arithmetic encoding engine (informative)	243
9.3.4.2	Encoding process for a binary decision (informative)	243
9.3.4.3	Renormalization process in the arithmetic encoding engine (informative).....	244
9.3.4.4	Bypass encoding process for binary decisions (informative).....	246
9.3.4.5	Encoding process for a binary decision before termination (informative).....	246
9.3.4.6	Byte stuffing process (informative)	247
Annex A	Profiles and levels.....	249
A.1	<i>Requirements on video decoder capability</i>	<i>249</i>
A.2	<i>Profiles.....</i>	<i>249</i>
A.2.1	Baseline profile	249
A.2.2	Main profile	250
A.2.3	Extended profile.....	250
A.2.4	High profile.....	250
A.2.5	High 10 profile.....	251
A.2.6	High 4:2:2 profile.....	251
A.2.7	High 4:4:4 profile.....	252

A.3	<i>Levels</i>	252
A.3.1	Level limits common to the Baseline, Main, and Extended profiles.....	252
A.3.2	Level limits common to the High, High 10, High 4:2:2, and High 4:4:4 profiles.....	254
A.3.3	Profile-specific level limits	255
A.3.3.1	Baseline profile limits	256
A.3.3.2	Main, High, High 10, High 4:2:2, or High 4:4:4 profile limits	257
A.3.3.3	Extended Profile Limits	258
A.3.4	Effect of level limits on frame rate (informative)	259
Annex B	Byte stream format	262
B.1	<i>Byte stream NAL unit syntax and semantics</i>	262
B.1.1	Byte stream NAL unit syntax	262
B.1.2	Byte stream NAL unit semantics	262
B.2	<i>Byte stream NAL unit decoding process</i>	262
B.3	<i>Decoder byte-alignment recovery (informative)</i>	263
Annex C	Hypothetical reference decoder	264
C.1	<i>Operation of coded picture buffer (CPB)</i>	266
C.1.1	Timing of bitstream arrival	266
C.1.2	Timing of coded picture removal	267
C.2	<i>Operation of the decoded picture buffer (DPB)</i>	268
C.2.1	Decoding of gaps in frame_num and storage of "non-existing" frames.....	268
C.2.2	Picture decoding and output.....	268
C.2.3	Removal of pictures from the DPB before possible insertion of the current picture	268
C.2.4	Current decoded picture marking and storage	269
C.2.4.1	Marking and storage of a reference decoded picture into the DPB	269
C.2.4.2	Storage of a non-reference picture into the DPB	269
C.3	<i>Bitstream conformance</i>	269
C.4	<i>Decoder conformance</i>	271
C.4.1	Operation of the output order DPB	272
C.4.2	Decoding of gaps in frame_num and storage of "non-existing" pictures.....	272
C.4.3	Picture decoding	272
C.4.4	Removal of pictures from the DPB before possible insertion of the current picture	272
C.4.5	Current decoded picture marking and storage	272
C.4.5.1	Storage and marking of a reference decoded picture into the DPB	272
C.4.5.2	Storage and marking of a non-reference decoded picture into the DPB.....	273
C.4.5.3	"Bumping" process.....	273
Annex D	Supplemental enhancement information	275
D.1	<i>SEI payload syntax</i>	276
D.1.1	Buffering period SEI message syntax	277
D.1.2	Picture timing SEI message syntax	277
D.1.3	Pan-scan rectangle SEI message syntax.....	278
D.1.4	Filler payload SEI message syntax	278
D.1.5	User data registered by ITU-T Recommendation T.35 SEI message syntax	279
D.1.6	User data unregistered SEI message syntax.....	279
D.1.7	Recovery point SEI message syntax	279
D.1.8	Decoded reference picture marking repetition SEI message syntax	279
D.1.9	Spare picture SEI message syntax	280
D.1.10	Scene information SEI message syntax	280
D.1.11	Sub-sequence information SEI message syntax.....	281
D.1.12	Sub-sequence layer characteristics SEI message syntax.....	281
D.1.13	Sub-sequence characteristics SEI message syntax.....	281
D.1.14	Full-frame freeze SEI message syntax.....	282
D.1.15	Full-frame freeze release SEI message syntax.....	282
D.1.16	Full-frame snapshot SEI message syntax.....	282
D.1.17	Progressive refinement segment start SEI message syntax.....	282
D.1.18	Progressive refinement segment end SEI message syntax.....	282
D.1.19	Motion-constrained slice group set SEI message syntax	282
D.1.20	Film grain characteristics SEI message syntax	283
D.1.21	Deblocking filter display preference SEI message syntax	283
D.1.22	Stereo video information SEI message syntax.....	284
D.1.23	Reserved SEI message syntax.....	284
D.2	<i>SEI payload semantics</i>	284
D.2.1	Buffering period SEI message semantics.....	284

D.2.2	Picture timing SEI message semantics.....	285
D.2.3	Pan-scan rectangle SEI message semantics	288
D.2.4	Filler payload SEI message semantics	290
D.2.5	User data registered by ITU-T Recommendation T.35 SEI message semantics.....	290
D.2.6	User data unregistered SEI message semantics	290
D.2.7	Recovery point SEI message semantics.....	290
D.2.8	Decoded reference picture marking repetition SEI message semantics.....	291
D.2.9	Spare picture SEI message semantics	292
D.2.10	Scene information SEI message semantics.....	293
D.2.11	Sub-sequence information SEI message semantics	295
D.2.12	Sub-sequence layer characteristics SEI message semantics.....	296
D.2.13	Sub-sequence characteristics SEI message semantics	297
D.2.14	Full-frame freeze SEI message semantics.....	299
D.2.15	Full-frame freeze release SEI message semantics.....	299
D.2.16	Full-frame snapshot SEI message semantics	299
D.2.17	Progressive refinement segment start SEI message semantics	299
D.2.18	Progressive refinement segment end SEI message semantics.....	300
D.2.19	Motion-constrained slice group set SEI message semantics	300
D.2.20	Film grain characteristics SEI message semantics	301
D.2.21	Deblocking filter display preference SEI message semantics.....	306
D.2.22	Stereo video information SEI message semantics	308
D.2.23	Reserved SEI message semantics	309
Annex E Video usability information.....		310
E.1	<i>VUI syntax</i>	<i>311</i>
E.1.1	VUI parameters syntax	311
E.1.2	HRD parameters syntax	312
E.2	<i>VUI semantics.....</i>	<i>312</i>
E.2.1	VUI parameters semantics	312
E.2.2	HRD parameters semantics.....	323

LIST OF FIGURES

Figure 6-1 – Nominal vertical and horizontal locations of 4:2:0 luma and chroma samples in a frame.....	19
Figure 6-2 – Nominal vertical and horizontal sampling locations of 4:2:0 samples in top and bottom fields.....	20
Figure 6-3 – Nominal vertical and horizontal locations of 4:2:2 luma and chroma samples in a frame.....	20
Figure 6-4 – Nominal vertical and horizontal sampling locations of 4:2:2 samples top and bottom fields.....	21
Figure 6-5 – Nominal vertical and horizontal locations of 4:4:4 luma and chroma samples in a frame.....	21
Figure 6-6 – Nominal vertical and horizontal sampling locations of 4:4:4 samples top and bottom fields.....	22
Figure 6-7 – A picture with 11 by 9 macroblocks that is partitioned into two slices.....	23
Figure 6-8 – Partitioning of the decoded frame into macroblock pairs	23
Figure 6-9 – Macroblock partitions, sub-macroblock partitions, macroblock partition scans, and sub-macroblock partition scans	25
Figure 6-10 – Scan for 4x4 luma blocks.....	26
Figure 6-11 – Scan for 8x8 luma blocks.....	26
Figure 6-12 – Neighbouring macroblocks for a given macroblock	27
Figure 6-13 – Neighbouring macroblocks for a given macroblock in MBAFF frames.....	28
Figure 6-14 – Determination of the neighbouring macroblock, blocks, and partitions (informative)	29
Figure 7-1 – Structure of an access unit not containing any NAL units with nal_unit_type equal to 0, 7, 8, or in the range of 12 to 18, inclusive, or in the range of 20 to 31, inclusive.	62
Figure 8-1 – Intra_4x4 prediction mode directions (informative)	118
Figure 8-2 –Example for temporal direct-mode motion vector inference (informative)	146
Figure 8-3 – Directional segmentation prediction (informative)	147

Figure 8-4 – Integer samples (shaded blocks with upper-case letters) and fractional sample positions (un-shaded blocks with lower-case letters) for quarter sample luma interpolation.....	153
Figure 8-5 – Fractional sample position dependent variables in chroma interpolation and surrounding integer position samples A, B, C, and D.....	155
Figure 8-6 – Assignment of the indices of dcY to luma4x4BlkIdx	161
Figure 8-7 – Assignment of the indices of dcC to chroma4x4BlkIdx: (a) chroma_format_idc equal to 1, (b) chroma_format_idc equal to 2, (c) chroma_format_idc equal to 3.....	163
Figure 8-8 – 4x4 block scans. (a) Zig-zag scan. (b) Field scan (informative)	164
Figure 8-9 – 8x8 block scans. (a) 8x8 zig-zag scan. (b) 8x8 field scan (informative).....	165
Figure 8-10 – Boundaries in a macroblock to be filtered.....	183
Figure 8-11 – Convention for describing samples across a 4x4 block horizontal or vertical boundary	187
Figure 9-1 – Illustration of CABAC parsing process for a syntax element SE (informative)	207
Figure 9-2 – Overview of the arithmetic decoding process for a single bin (informative)	238
Figure 9-3 – Flowchart for decoding a decision	239
Figure 9-4 – Flowchart of renormalization	241
Figure 9-5 – Flowchart of bypass decoding process.....	242
Figure 9-6 – Flowchart of decoding a decision before termination	243
Figure 9-7 – Flowchart for encoding a decision	244
Figure 9-8 – Flowchart of renormalization in the encoder	245
Figure 9-9 – Flowchart of PutBit(B).....	245
Figure 9-10 – Flowchart of encoding bypass.....	246
Figure 9-11 – Flowchart of encoding a decision before termination	247
Figure 9-12 – Flowchart of flushing at termination.....	247
Figure C-1 – Structure of byte streams and NAL unit streams for HRD conformance checks	264
Figure C-2 – HRD buffer model.....	265
Figure E-1 – Location of chroma samples for top and bottom fields as a function of chroma_sample_loc_type_top_field and chroma_sample_loc_type_bottom_field	320

LIST OF TABLES

Table 6-1 –SubWidthC, and SubHeightC values derived from chroma_format_idc.....	18
Table 6-2 – Specification of input and output assignments for subclauses 6.4.8.1 to 6.4.8.5.....	29
Table 6-3 – Specification of mbAddrN	33
Table 6-4 - Specification of mbAddrN and yM.....	35
Table 7-1 – NAL unit type codes.....	57
Table 7-2 – Assignment of mnemonic names to scaling list indices and specification of fall-back rule.....	65
Table 7-3 – Specification of default scaling lists Default_4x4_Intra and Default_4x4_Inter.....	66
Table 7-4 – Specification of default scaling lists Default_8x8_Intra and Default_8x8_Inter.....	66
Table 7-5 – Meaning of primary_pic_type	73
Table 7-6 – Name association to slice_type	75
Table 7-7 – reordering_of_pic_nums_idc operations for reordering of reference picture lists.....	81
Table 7-8 – Interpretation of adaptive_ref_pic_marking_mode_flag	83

Table 7-9 – Memory management control operation (memory_management_control_operation) values	84
Table 7-10 – Allowed collective macroblock types for slice_type	86
Table 7-11 – Macroblock types for I slices	87
Table 7-12 – Macroblock type with value 0 for SI slices	88
Table 7-13 – Macroblock type values 0 to 4 for P and SP slices	89
Table 7-14 – Macroblock type values 0 to 22 for B slices.....	90
Table 7-15 – Specification of CodedBlockPatternChroma values.....	92
Table 7-16 – Relationship between intra_chroma_pred_mode and spatial prediction modes.....	92
Table 7-17 – Sub-macroblock types in P macroblocks.....	93
Table 7-18 – Sub-macroblock types in B macroblocks	94
Table 8-1 – Refined slice group map type	102
Table 8-2 – Specification of Intra4x4PredMode[luma4x4BlkIdx] and associated names	117
Table 8-3 – Specification of Intra8x8PredMode[luma8x8BlkIdx] and associated names	123
Table 8-4 – Specification of Intra16x16PredMode and associated names	130
Table 8-5 – Specification of Intra chroma prediction modes and associated names.....	132
Table 8-6 – Specification of the variable colPic	140
Table 8-7 – Specification of PicCodingStruct(X).....	140
Table 8-8 – Specification of mbAddrCol, yM, and vertMvScale	141
Table 8-9 – Assignment of prediction utilization flags.....	143
Table 8-10 – Derivation of the vertical component of the chroma vector in field coding mode	149
Table 8-11 – Differential full-sample luma locations	153
Table 8-12 – Assignment of the luma prediction sample predPartLX _L [x _L , y _L]	155
Table 8-13 – Specification of mapping of idx to c _{ij} for zig-zag and field scan.....	164
Table 8-14 – Specification of mapping of idx to c _{ij} for 8x8 zig-zag and 8x8 field scan.....	166
Table 8-15 – Specification of QP _C as a function of qP _I	167
Table 8-16 – Derivation of offset dependent threshold variables α' and β' from indexA and indexB.....	191
Table 8-17 – Value of variable t' _{C0} as a function of indexA and bS	192
Table 9-1 – Bit strings with “prefix” and “suffix” bits and assignment to codeNum ranges (informative).....	194
Table 9-2 – Exp-Golomb bit strings and codeNum in explicit form and used as ue(v) (informative).....	195
Table 9-3 – Assignment of syntax element to codeNum for signed Exp-Golomb coded syntax elements se(v).....	196
Table 9-4 – Assignment of codeNum to values of coded_block_pattern for macroblock prediction modes	196
Table 9-5 – coeff_token mapping to TotalCoeff(coeff_token) and TrailingOnes(coeff_token).....	200
Table 9-6 – Codeword table for level_prefix (informative).....	203
Table 9-7 – total_zeros tables for 4x4 blocks with TotalCoeff(coeff_token) 1 to 7	204
Table 9-8 – total_zeros tables for 4x4 blocks with TotalCoeff(coeff_token) 8 to 15	205
Table 9-9 – total_zeros tables for chroma DC 2x2 and 2x4 blocks	205
Table 9-10 – Tables for run_before	206
Table 9-11 – Association of ctxIdx and syntax elements for each slice type in the initialisation process.....	208
Table 9-12 – Values of variables m and n for ctxIdx from 0 to 10.....	209
Table 9-13 – Values of variables m and n for ctxIdx from 11 to 23.....	210

Table 9-14 – Values of variables m and n for ctxIdx from 24 to 39.....	210
Table 9-15 – Values of variables m and n for ctxIdx from 40 to 53.....	210
Table 9-16 – Values of variables m and n for ctxIdx from 54 to 59, and 399 to 401	211
Table 9-17 – Values of variables m and n for ctxIdx from 60 to 69.....	211
Table 9-18 – Values of variables m and n for ctxIdx from 70 to 104.....	212
Table 9-19 – Values of variables m and n for ctxIdx from 105 to 165.....	213
Table 9-20 – Values of variables m and n for ctxIdx from 166 to 226.....	214
Table 9-21 – Values of variables m and n for ctxIdx from 227 to 275.....	215
Table 9-22 – Values of variables m and n for ctxIdx from 277 to 337.....	216
Table 9-23 – Values of variables m and n for ctxIdx from 338 to 398.....	217
Table 9-24 – Values of variables m and n for ctxIdx from 402 to 459.....	218
Table 9-25 – Syntax elements and associated types of binarization, maxBinIdxCtx, and ctxIdxOffset.....	220
Table 9-26 – Bin string of the unary binarization (informative).....	221
Table 9-27 – Binarization for macroblock types in I slices	223
Table 9-28 – Binarization for macroblock types in P, SP, and B slices.....	224
Table 9-29 – Binarization for sub-macroblock types in P, SP, and B slices.....	225
Table 9-30 – Assignment of ctxIdxInc to binIdx for all ctxIdxOffset values except those related to the syntax elements coded_block_flag, significant_coeff_flag, last_significant_coeff_flag, and coeff_abs_level_minus1.....	227
Table 9-31 – Assignment of ctxIdxBlockCatOffset to ctxBlockCat for syntax elements coded_block_flag, significant_coeff_flag, last_significant_coeff_flag, and coeff_abs_level_minus1	228
Table 9-32 – Specification of ctxIdxInc for specific values of ctxIdxOffset and binIdx.....	235
Table 9-33 – Specification of ctxBlockCat for the different blocks	235
Table 9-34 – Mapping of scanning position to ctxIdxInc for ctxBlockCat == 5	236
Table 9-35 – Specification of rangeTabLPS depending on pStateIdx and qCodIRangeIdx	240
Table 9-36 – State transition table	241
Table A-1 – Level limits.....	254
Table A-2 – Specification of cpbBrVclFactor and cpbBrNalFactor.....	256
Table A-3 – Baseline profile level limits.....	257
Table A-4 – Main, High, High 10, High 4:2:2, or High 4:4:4 profile level limits	257
Table A-5 – Extended profile level limits.....	258
Table A-6 – Maximum frame rates (frames per second) for some example frame sizes.....	259
Table D-1 – Interpretation of pic_struct	286
Table D-2 – Mapping of ct_type to source picture scan.....	287
Table D-3 – Definition of counting_type values	287
Table D-4 – scene_transition_type values.	294
Table D-5 – model_id values.....	301
Table D-6 – blending_mode_id values.....	302
Table E-1 – Meaning of sample aspect ratio indicator	313
Table E-2 – Meaning of video_format.....	314
Table E-3 – Colour primaries	315

Table E-4 – Transfer characteristics	316
Table E-5 – Matrix coefficients	319
Table E-6 – Divisor for computation of $\Delta t_{fi,dpb}(n)$	321

Foreword

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardising telecommunications on a world-wide basis. The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups that, in turn, produce Recommendations on these topics. The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1. In some areas of information technology that fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialised system for world-wide standardisation. National Bodies that are members of ISO and IEC participate in the development of International Standards through technical committees established by the respective organisation to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organisations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

This Recommendation | International Standard was prepared jointly by ITU-T SG 16 Q.6, also known as VCEG (Video Coding Experts Group), and by ISO/IEC JTC 1/SC 29/WG 11, also known as MPEG (Moving Picture Experts Group). VCEG was formed in 1997 to maintain prior ITU-T video coding standards and develop new video coding standard(s) appropriate for a wide range of conversational and non-conversational services. MPEG was formed in 1988 to establish standards for coding of moving pictures and associated audio for various applications such as digital storage media, distribution, and communication.

In this Recommendation | International Standard Annexes A through E contain normative requirements and are an integral part of this Recommendation | International Standard.

ITU-T Recommendation H.264

Advanced video coding for generic audiovisual services

0 Introduction

This clause does not form an integral part of this Recommendation | International Standard.

0.1 Prologue

This subclause does not form an integral part of this Recommendation | International Standard.

As the costs for both processing power and memory have reduced, network support for coded video data has diversified, and advances in video coding technology have progressed, the need has arisen for an industry standard for compressed video representation with substantially increased coding efficiency and enhanced robustness to network environments. Toward these ends the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) formed a Joint Video Team (JVT) in 2001 for development of a new Recommendation | International Standard.

0.2 Purpose

This subclause does not form an integral part of this Recommendation | International Standard.

This Recommendation | International Standard was developed in response to the growing need for higher compression of moving pictures for various applications such as videoconferencing, digital storage media, television broadcasting, internet streaming, and communication. It is also designed to enable the use of the coded video representation in a flexible manner for a wide variety of network environments. The use of this Recommendation | International Standard allows motion video to be manipulated as a form of computer data and to be stored on various storage media, transmitted and received over existing and future networks and distributed on existing and future broadcasting channels.

0.3 Applications

This subclause does not form an integral part of this Recommendation | International Standard.

This Recommendation | International Standard is designed to cover a broad range of applications for video content including but not limited to the following:

CATV	Cable TV on optical networks, copper, etc.
DBS	Direct broadcast satellite video services
DSL	Digital subscriber line video services
DTTB	Digital terrestrial television broadcasting
ISM	Interactive storage media (optical disks, etc.)
MMM	Multimedia mailing
MSPN	Multimedia services over packet networks
RTC	Real-time conversational services (videoconferencing, videophone, etc.)
RVS	Remote video surveillance
SSM	Serial storage media (digital VTR, etc.)

0.4 Publication and versions of this specification

This subclause does not form an integral part of this Recommendation | International Standard.

This specification has been jointly developed by ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group. It is published as technically-aligned twin text in both organizations ITU-T and ISO/IEC.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 1 refers to the first (2003) approved version of this Recommendation | International Standard.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 2 refers to the integrated text containing the corrections specified in the first technical corrigendum.

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 3 refers to the integrated text containing both the first technical corrigendum (2004) and the first amendment, which is referred to as the "Fidelity range extensions".

ITU-T Rec. H.264 | ISO/IEC 14496-10 version 4 (the current specification) refers to the integrated text containing the first technical corrigendum (2004), the first amendment (the "Fidelity range extensions"), and an additional technical corrigendum (2005). In the ITU-T, the next published version after version 2 was version 4 (due to the completion of the drafting work for version 4 prior to the approval opportunity for a final version 3 text).

0.5 Profiles and levels

This subclause does not form an integral part of this Recommendation | International Standard.

This Recommendation | International Standard is designed to be generic in the sense that it serves a wide range of applications, bit rates, resolutions, qualities, and services. Applications should cover, among other things, digital storage media, television broadcasting and real-time communications. In the course of creating this Specification, various requirements from typical applications have been considered, necessary algorithmic elements have been developed, and these have been integrated into a single syntax. Hence, this Specification will facilitate video data interchange among different applications.

Considering the practicality of implementing the full syntax of this Specification, however, a limited number of subsets of the syntax are also stipulated by means of "profiles" and "levels". These and other related terms are formally defined in clause 3.

A "profile" is a subset of the entire bitstream syntax that is specified by this Recommendation | International Standard. Within the bounds imposed by the syntax of a given profile it is still possible to require a very large variation in the performance of encoders and decoders depending upon the values taken by syntax elements in the bitstream such as the specified size of the decoded pictures. In many applications, it is currently neither practical nor economic to implement a decoder capable of dealing with all hypothetical uses of the syntax within a particular profile.

In order to deal with this problem, "levels" are specified within each profile. A level is a specified set of constraints imposed on values of the syntax elements in the bitstream. These constraints may be simple limits on values. Alternatively they may take the form of constraints on arithmetic combinations of values (e.g. picture width multiplied by picture height multiplied by number of pictures decoded per second).

Coded video content conforming to this Recommendation | International Standard uses a common syntax. In order to achieve a subset of the complete syntax, flags, parameters, and other syntax elements are included in the bitstream that signal the presence or absence of syntactic elements that occur later in the bitstream.

0.6 Overview of the design characteristics

This subclause does not form an integral part of this Recommendation | International Standard.

The coded representation specified in the syntax is designed to enable a high compression capability for a desired image quality. With the exception of the transform bypass mode of operation for lossless coding in the High 4:4:4 profile and the I_PCM mode of operation in all profiles, the algorithm is typically not lossless, as the exact source sample values are typically not preserved through the encoding and decoding processes. A number of techniques may be used to achieve highly efficient compression. Encoding algorithms (not specified in this Recommendation | International Standard) may select between inter and intra coding for block-shaped regions of each picture. Inter coding uses motion vectors for block-based inter prediction to exploit temporal statistical dependencies between different pictures. Intra coding uses various spatial prediction modes to exploit spatial statistical dependencies in the source signal for a single picture. Motion vectors and intra prediction modes may be specified for a variety of block sizes in the picture. The prediction residual is then further compressed using a transform to remove spatial correlation inside the transform block before it is quantised, producing an irreversible process that typically discards less important visual information while forming a close approximation to the source samples. Finally, the motion vectors or intra prediction modes are combined with the quantised transform coefficient information and encoded using either variable length codes or arithmetic coding.

0.6.1 Predictive coding

This subclause does not form an integral part of this Recommendation | International Standard.

Because of the conflicting requirements of random access and highly efficient compression, two main coding types are specified. Intra coding is done without reference to other pictures. Intra coding may provide access points to the coded sequence where decoding can begin and continue correctly, but typically also shows only moderate compression efficiency. Inter coding (predictive or bi-predictive) is more efficient using inter prediction of each block of sample values from some previously decoded picture selected by the encoder. In contrast to some other video coding standards, pictures coded using bi-predictive inter prediction may also be used as references for inter coding of other pictures.

The application of the three coding types to pictures in a sequence is flexible, and the order of the decoding process is generally not the same as the order of the source picture capture process in the encoder or the output order from the decoder for display. The choice is left to the encoder and will depend on the requirements of the application. The decoding order is specified such that the decoding of pictures that use inter-picture prediction follows later in decoding order than other pictures that are referenced in the decoding process.

0.6.2 Coding of progressive and interlaced video

This subclause does not form an integral part of this Recommendation | International Standard.

This Recommendation | International Standard specifies a syntax and decoding process for video that originated in either progressive-scan or interlaced-scan form, which may be mixed together in the same sequence. The two fields of an interlaced frame are separated in capture time while the two fields of a progressive frame share the same capture time. Each field may be coded separately or the two fields may be coded together as a frame. Progressive frames are typically coded as a frame. For interlaced video, the encoder can choose between frame coding and field coding. Frame coding or field coding can be adaptively selected on a picture-by-picture basis and also on a more localized basis within a coded frame. Frame coding is typically preferred when the video scene contains significant detail with limited motion. Field coding typically works better when there is fast picture-to-picture motion.

0.6.3 Picture partitioning into macroblocks and smaller partitions

This subclause does not form an integral part of this Recommendation | International Standard.

As in previous video coding Recommendations and International Standards, a macroblock, consisting of a 16x16 block of luma samples and two corresponding blocks of chroma samples, is used as the basic processing unit of the video decoding process.

A macroblock can be further partitioned for inter prediction. The selection of the size of inter prediction partitions is a result of a trade-off between the coding gain provided by using motion compensation with smaller blocks and the quantity of data needed to represent the data for motion compensation. In this Recommendation | International Standard the inter prediction process can form segmentations for motion representation as small as 4x4 luma samples in size, using motion vector accuracy of one-quarter of the luma sample grid spacing displacement. The process for inter prediction of a sample block can also involve the selection of the picture to be used as the reference picture from a number of stored previously-decoded pictures. Motion vectors are encoded differentially with respect to predicted values formed from nearby encoded motion vectors.

Typically, the encoder calculates appropriate motion vectors and other data elements represented in the video data stream. This motion estimation process in the encoder and the selection of whether to use inter prediction for the representation of each region of the video content is not specified in this Recommendation | International Standard.

0.6.4 Spatial redundancy reduction

This subclause does not form an integral part of this Recommendation | International Standard.

Both source pictures and prediction residuals have high spatial redundancy. This Recommendation | International Standard is based on the use of a block-based transform method for spatial redundancy removal. After inter prediction from previously-decoded samples in other pictures or spatial-based prediction from previously-decoded samples within the current picture, the resulting prediction residual is split into 4x4 blocks. These are converted into the transform domain where they are quantised. After quantisation many of the transform coefficients are zero or have low amplitude and can thus be represented with a small amount of encoded data. The processes of transformation and quantisation in the encoder are not specified in this Recommendation | International Standard.

0.7 How to read this specification

This subclause does not form an integral part of this Recommendation | International Standard.

It is suggested that the reader starts with clause 1 (Scope) and moves on to clause 3 (Definitions). Clause 6 should be read for the geometrical relationship of the source, input, and output of the decoder. Clause 7 (Syntax and semantics) specifies the order to parse syntax elements from the bitstream. See subclauses 7.1-7.3 for syntactical order and see subclause 7.4 for semantics; i.e., the scope, restrictions, and conditions that are imposed on the syntax elements. The actual parsing for most syntax elements is specified in clause 9 (Parsing process). Finally, clause 8 (Decoding process) specifies how the syntax elements are mapped into decoded samples. Throughout reading this specification, the reader should refer to clauses 2 (Normative references), 4 (Abbreviations), and 5 (Conventions) as needed. Annexes A through E also form an integral part of this Recommendation | International Standard.

Annex A specifies seven profiles (Baseline, Main, Extended, High, High 10, High 4:2:2 and High 4:4:4), each being tailored to certain application domains, and defines the so-called levels of the profiles. Annex B specifies syntax and semantics of a byte stream format for delivery of coded video as an ordered stream of bytes. Annex C specifies the hypothetical reference decoder and its use to check bitstream and decoder conformance. Annex D specifies syntax and semantics for supplemental enhancement information message payloads. Finally, Annex E specifies syntax and semantics of the video usability information parameters of the sequence parameter set.

Throughout this specification, statements appearing with the preamble "NOTE -" are informative and are not an integral part of this Recommendation | International Standard.

1 Scope

This document specifies ITU-T Recommendation H.264 | ISO/IEC International Standard ISO/IEC 14496-10 video coding.

2 Normative references

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

- ITU-T Recommendation T.35 (2000), *Procedure for the allocation of ITU-T defined codes for non-standard facilities*.
- ISO/IEC 11578:1996, Annex A, *Universal Unique Identifier*.
- ISO/CIE 10527:1991, *Colorimetric Observers*.

3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply.

- 3.1 access unit:** A set of *NAL units* always containing exactly one *primary coded picture*. In addition to the *primary coded picture*, an access unit may also contain one or more *redundant coded pictures* or other *NAL units* not containing *slices* or *slice data partitions* of a *coded picture*. The decoding of an access unit always results in a *decoded picture*.
- 3.2 AC transform coefficient:** Any *transform coefficient* for which the *frequency index* in one or both dimensions is non-zero.
- 3.3 adaptive binary arithmetic decoding process:** An entropy *decoding process* that derives the values of *bins* from a *bitstream* produced by an *adaptive binary arithmetic encoding process*.
- 3.4 adaptive binary arithmetic encoding process:** An entropy *encoding process*, not normatively specified in this Recommendation | International Standard, that codes a sequence of *bins* and produces a *bitstream* that can be decoded using the *adaptive binary arithmetic decoding process*.
- 3.5 alpha blending:** A process not specified by this Recommendation | International Standard, in which an *auxiliary coded picture* is used in combination with a *primary coded picture* and with other data not specified by this Recommendation | International Standard in the *display process*. In an alpha blending process, the samples of an *auxiliary coded picture* are interpreted as indications of the degree of opacity (or, equivalently, the degrees of transparency) associated with the corresponding *luma* samples of the *primary coded picture*.

- 3.6 arbitrary slice order:** A *decoding order* of *slices* in which the *macroblock address* of the first *macroblock* of some *slice* of a *picture* may be less than the *macroblock address* of the first *macroblock* of some other preceding *slice* of the same *coded picture*.
- 3.7 auxiliary coded picture:** A *picture* that supplements the *primary coded picture* that may be used in combination with other data not specified by this Recommendation | International Standard in the *display process*. An auxiliary coded picture has the same syntactic and semantic restrictions as a monochrome *redundant coded picture*. An auxiliary coded picture must contain the same number of *macroblocks* as the *primary coded picture*. Auxiliary coded pictures have no normative effect on the *decoding process*. See also *primary coded picture* and *redundant coded picture*.
- 3.8 B slice:** A *slice* that may be decoded using *intra prediction* from decoded samples within the same *slice* or *inter prediction* from previously-decoded *reference pictures*, using at most two *motion vectors* and *reference indices* to *predict* the sample values of each *block*.
- 3.9 bin:** One bit of a *bin string*.
- 3.10 binarization:** A set of *bin strings* for all possible values of a *syntax element*.
- 3.11 binarization process:** A unique mapping process of all possible values of a *syntax element* onto a set of *bin strings*.
- 3.12 bin string:** A string of *bins*. A bin string is an intermediate binary representation of values of *syntax elements* from the *binarization* of the *syntax element*.
- 3.13 bi-predictive slice:** See *B slice*.
- 3.14 bitstream:** A sequence of bits that forms the representation of *coded pictures* and associated data forming one or more *coded video sequences*. Bitstream is a collective term used to refer either to a *NAL unit stream* or a *byte stream*.
- 3.15 block:** An MxN (M-column by N-row) array of samples, or an MxN array of *transform coefficients*.
- 3.16 bottom field:** One of two *fields* that comprise a *frame*. Each row of a *bottom field* is spatially located immediately below a corresponding row of a *top field*.
- 3.17 bottom macroblock (of a macroblock pair):** The *macroblock* within a *macroblock pair* that contains the samples in the bottom row of samples for the *macroblock pair*. For a *field macroblock pair*, the bottom macroblock represents the samples from the region of the *bottom field* of the *frame* that lie within the spatial region of the *macroblock pair*. For a *frame macroblock pair*, the bottom macroblock represents the samples of the *frame* that lie within the bottom half of the spatial region of the *macroblock pair*.
- 3.18 broken link:** A location in a *bitstream* at which it is indicated that some subsequent *pictures* in *decoding order* may contain serious visual artefacts due to unspecified operations performed in the generation of the *bitstream*.
- 3.19 byte:** A sequence of 8 bits, written and read with the most significant bit on the left and the least significant bit on the right. When represented in a sequence of data bits, the most significant bit of a byte is first.
- 3.20 byte-aligned:** A position in a *bitstream* is byte-aligned when the position is an integer multiple of 8 bits from the position of the first bit in the *bitstream*. A bit or *byte* or *syntax element* is said to be byte-aligned when the position at which it appears in a *bitstream* is byte-aligned.
- 3.21 byte stream:** An encapsulation of a *NAL unit stream* containing *start code prefixes* and *NAL units* as specified in Annex B.
- 3.22 can:** A term used to refer to behaviour that is allowed, but not necessarily required.
- 3.23 category:** A number associated with each *syntax element*. The category is used to specify the allocation of *syntax elements* to *NAL units* for *slice data partitioning*. It may also be used in a manner determined by the application to refer to classes of *syntax elements* in a manner not specified in this Recommendation | International Standard.
- 3.24 chroma:** An adjective specifying that a sample array or single sample is representing one of the two colour difference signals related to the primary colours. The symbols used for a chroma array or sample are Cb and Cr.
- NOTE – The term chroma is used rather than the term chrominance in order to avoid the implication of the use of linear light transfer characteristics that is often associated with the term chrominance.
- 3.25 coded field:** A *coded representation* of a *field*.

- 3.26 coded frame:** A *coded representation* of a *frame*.
- 3.27 coded picture:** A *coded representation* of a *picture*. A coded picture may be either a *coded field* or a *coded frame*. Coded picture is a collective term referring to a *primary coded picture* or a *redundant coded picture*, but not to both together.
- 3.28 coded picture buffer (CPB):** A first-in first-out buffer containing *access units* in *decoding order* specified in the *hypothetical reference decoder* in Annex C.
- 3.29 coded representation:** A data element as represented in its coded form.
- 3.30 coded video sequence:** A sequence of *access units* that consists, in decoding order, of an *IDR access unit* followed by zero or more non-IDR *access units* including all subsequent *access units* up to but not including any subsequent *IDR access unit*.
- 3.31 component:** An array or single sample from one of the three arrays (*luma* and two *chroma*) that make up a *field* or *frame*.
- 3.32 complementary field pair:** A collective term for a *complementary reference field pair* or a *complementary non-reference field pair*.
- 3.33 complementary non-reference field pair:** Two *non-reference fields* that are in consecutive *access units* in *decoding order* as two *coded fields* of opposite parity where the first *field* is not already a paired *field*.
- 3.34 complementary reference field pair:** Two *reference fields* that are in consecutive *access units* in *decoding order* as two *coded fields* and share the same value of the *frame_num syntax element*, where the second *field* in *decoding order* is not an *IDR picture* and does not include a *memory_management_control_operation syntax element* equal to 5.
- 3.35 context variable:** A variable specified for the *adaptive binary arithmetic decoding process* of a *bin* by an equation containing recently decoded *bins*.
- 3.36 DC transform coefficient:** A *transform coefficient* for which the *frequency index* is zero in all dimensions.
- 3.37 decoded picture:** A *decoded picture* is derived by decoding a *coded picture*. A *decoded picture* is either a *decoded frame*, or a *decoded field*. A *decoded field* is either a *decoded top field* or a *decoded bottom field*.
- 3.38 decoded picture buffer (DPB):** A buffer holding *decoded pictures* for reference, output reordering, or output delay specified for the *hypothetical reference decoder* in Annex C.
- 3.39 decoder:** An embodiment of a *decoding process*.
- 3.40 decoding order:** The order in which *syntax elements* are processed by the *decoding process*.
- 3.41 decoding process:** The process specified in this Recommendation | International Standard that reads a *bitstream* and derives *decoded pictures* from it.
- 3.42 direct prediction:** An *inter prediction* for a *block* for which no *motion vector* is decoded. Two direct prediction modes are specified that are referred to as *spatial direct prediction* and *temporal prediction mode*.
- 3.43 display process:** A process not specified in this Recommendation | International Standard having, as its input, the cropped *decoded pictures* that are the output of the *decoding process*.
- 3.44 decoder under test (DUT):** A *decoder* that is tested for conformance to this Recommendation | International Standard by operating the *hypothetical stream scheduler* to deliver a conforming *bitstream* to the *decoder* and to the *hypothetical reference decoder* and comparing the values and timing of the output of the two *decoders*.
- 3.45 emulation prevention byte:** A *byte* equal to 0x03 that may be present within a *NAL unit*. The presence of emulation prevention bytes ensures that no sequence of consecutive *byte-aligned bytes* in the *NAL unit* contains a *start code prefix*.
- 3.46 encoder:** An embodiment of an *encoding process*.
- 3.47 encoding process:** A process, not specified in this Recommendation | International Standard, that produces a *bitstream* conforming to this Recommendation | International Standard.
- 3.48 field:** An assembly of alternate rows of a *frame*. A *frame* is composed of two *fields*, a *top field* and a *bottom field*.
- 3.49 field macroblock:** A *macroblock* containing samples from a single *field*. All *macroblocks* of a *coded field* are *field macroblocks*. When *macroblock-adaptive frame/field decoding* is in use, some *macroblocks* of a *coded frame* may be *field macroblocks*.

- 3.50 field macroblock pair:** A *macroblock pair* decoded as two *field macroblocks*.
- 3.51 field scan:** A specific sequential ordering of *transform coefficients* that differs from the *zig-zag scan* by scanning columns more rapidly than rows. Field scan is used for *transform coefficients* in *field macroblocks*.
- 3.52 flag:** A variable that can take one of the two possible values 0 and 1.
- 3.53 frame:** A *frame* contains an array of *luma* samples and two corresponding arrays of *chroma* samples. A *frame* consists of two *fields*, a *top field* and a *bottom field*.
- 3.54 frame macroblock:** A *macroblock* representing samples from the two *fields* of a *coded frame*. When *macroblock-adaptive frame/field decoding* is not in use, all *macroblocks* of a *coded frame* are *frame macroblocks*. When *macroblock-adaptive frame/field decoding* is in use, some *macroblocks* of a *coded frame* may be *frame macroblocks*.
- 3.55 frame macroblock pair:** A *macroblock pair* decoded as two *frame macroblocks*.
- 3.56 frequency index:** A one-dimensional or two-dimensional index associated with a *transform coefficient* prior to an *inverse transform* part of the *decoding process*.
- 3.57 hypothetical reference decoder (HRD):** A hypothetical *decoder* model that specifies constraints on the variability of conforming *NAL unit streams* or conforming *byte streams* that an encoding process may produce.
- 3.58 hypothetical stream scheduler (HSS):** A hypothetical delivery mechanism for the timing and data flow of the input of a *bitstream* into the *hypothetical reference decoder*. The HSS is used for checking the conformance of a *bitstream* or a *decoder*.
- 3.59 I slice:** A *slice* that is not an *SI slice* that is decoded using *prediction* only from decoded samples within the same *slice*.
- 3.60 informative:** A term used to refer to content provided in this Recommendation | International Standard that is not an integral part of this Recommendation | International Standard. Informative content does not establish any mandatory requirements for conformance to this Recommendation | International Standard.
- 3.61 instantaneous decoding refresh (IDR) access unit:** An *access unit* in which the *primary coded picture* is an *IDR picture*.
- 3.62 instantaneous decoding refresh (IDR) picture:** A *coded picture* in which all *slices* are *I* or *SI slices* that causes the *decoding process* to mark all *reference pictures* as "unused for reference" immediately after decoding the *IDR picture*. After the decoding of an *IDR picture* all following *coded pictures* in *decoding order* can be decoded without *inter prediction* from any *picture* decoded prior to the *IDR picture*. The first *picture* of each *coded video sequence* is an *IDR picture*.
- 3.63 inter coding:** Coding of a *block*, *macroblock*, *slice*, or *picture* that uses *inter prediction*.
- 3.64 inter prediction:** A *prediction* derived from decoded samples of *reference pictures* other than the current decoded *picture*.
- 3.65 interpretation sample value:** A possibly-altered value corresponding to a decoded sample value of an *auxiliary coded picture* that may be generated for use in the *display process*. Interpretation sample values are not used in the *decoding process* and have no normative effect on the *decoding process*.
- 3.66 intra coding:** Coding of a *block*, *macroblock*, *slice*, or *picture* that uses *intra prediction*.
- 3.67 intra prediction:** A *prediction* derived from the decoded samples of the same decoded *slice*.
- 3.68 intra slice:** See *I slice*.
- 3.69 inverse transform:** A part of the *decoding process* by which a set of *transform coefficients* are converted into spatial-domain values, or by which a set of *transform coefficients* are converted into *DC transform coefficients*.
- 3.70 layer:** One of a set of syntactical structures in a non-branching hierarchical relationship. Higher layers contain lower layers. The coding layers are the *coded video sequence*, *picture*, *slice*, and *macroblock* layers.
- 3.71 level:** A defined set of constraints on the values that may be taken by the *syntax elements* and variables of this Recommendation | International Standard. The same set of levels is defined for all *profiles*, with most aspects of the definition of each level being in common across different *profiles*. Individual implementations may, within specified constraints, support a different level for each supported *profile*. In a different context, level is the value of a *transform coefficient* prior to *scaling*.

- 3.72 list 0 (list 1) motion vector:** A *motion vector* associated with a *reference index* pointing into *reference picture list 0 (list 1)*.
- 3.73 list 0 (list 1) prediction:** *Inter prediction* of the content of a *slice* using a *reference index* pointing into *reference picture list 0 (list 1)*.
- 3.74 luma:** An adjective specifying that a sample array or single sample is representing the monochrome signal related to the primary colours. The symbol or subscript used for luma is Y or L.
NOTE – The term luma is used rather than the term luminance in order to avoid the implication of the use of linear light transfer characteristics that is often associated with the term luminance. The symbol L is sometimes used instead of the symbol Y to avoid confusion with the symbol y as used for vertical location.
- 3.75 macroblock:** A 16x16 *block* of *luma* samples and two corresponding *blocks* of *chroma* samples. The division of a *slice* or a *macroblock pair* into macroblocks is a *partitioning*.
- 3.76 macroblock-adaptive frame/field decoding:** A *decoding process* for *coded frames* in which some *macroblocks* may be decoded as *frame macroblocks* and others may be decoded as *field macroblocks*.
- 3.77 macroblock address:** When *macroblock-adaptive frame/field decoding* is not in use, a macroblock address is the index of a macroblock in a *macroblock raster scan* of the *picture* starting with zero for the top-left *macroblock* in a *picture*. When *macroblock-adaptive frame/field decoding* is in use, the macroblock address of the *top macroblock* of a *macroblock pair* is two times the index of the *macroblock pair* in a *macroblock pair raster scan* of the *picture*, and the macroblock address of the *bottom macroblock* of a *macroblock pair* is the macroblock address of the corresponding *top macroblock* plus 1. The macroblock address of the *top macroblock* of each *macroblock pair* is an even number and the macroblock address of the *bottom macroblock* of each *macroblock pair* is an odd number.
- 3.78 macroblock location:** The two-dimensional coordinates of a *macroblock* in a *picture* denoted by (x, y). For the top left *macroblock* of the *picture* (x, y) is equal to (0, 0). x is incremented by 1 for each *macroblock* column from left to right. When *macroblock-adaptive frame/field decoding* is not in use, y is incremented by 1 for each *macroblock* row from top to bottom. When *macroblock-adaptive frame/field decoding* is in use, y is incremented by 2 for each *macroblock pair* row from top to bottom, and is incremented by an additional 1 when a macroblock is a *bottom macroblock*.
- 3.79 macroblock pair:** A pair of vertically contiguous *macroblocks* in a *frame* that is coupled for use in *macroblock-adaptive frame/field decoding*. The division of a *slice* into macroblock pairs is a *partitioning*.
- 3.80 macroblock partition:** A *block* of *luma* samples and two corresponding *blocks* of *chroma* samples resulting from a *partitioning* of a *macroblock* for *inter prediction*.
- 3.81 macroblock to slice group map:** A means of mapping *macroblocks* of a *picture* into *slice groups*. The macroblock to slice group map consists of a list of numbers, one for each coded *macroblock*, specifying the *slice group* to which each coded *macroblock* belongs.
- 3.82 map unit to slice group map:** A means of mapping *slice group map units* of a *picture* into *slice groups*. The map unit to slice group map consists of a list of numbers, one for each *slice group map unit*, specifying the *slice group* to which each coded *slice group map unit* belongs.
- 3.83 may:** A term used to refer to behaviour that is allowed, but not necessarily required. In some places where the optional nature of the described behaviour is intended to be emphasized, the phrase "may or may not" is used to provide emphasis.
- 3.84 memory management control operation:** Seven operations that control *reference picture marking*.
- 3.85 motion vector:** A two-dimensional vector used for *inter prediction* that provides an offset from the coordinates in the *decoded picture* to the coordinates in a *reference picture*.
- 3.86 must:** A term used in expressing an observation about a requirement or an implication of a requirement that is specified elsewhere in this Recommendation | International Standard. This term is used exclusively in an *informative* context.
- 3.87 NAL unit:** A syntax structure containing an indication of the type of data to follow and *bytes* containing that data in the form of an *RBSP* interspersed as necessary with *emulation prevention bytes*.
- 3.88 NAL unit stream:** A sequence of *NAL units*.
- 3.89 non-paired field:** A collective term for a *non-paired reference field* or a *non-paired non-reference field*.
- 3.90 non-paired non-reference field:** A decoded *non-reference field* that is not part of a *complementary non-reference field pair*.

- 3.91 non-paired reference field:** A decoded *reference field* that is not part of a *complementary reference field pair*.
- 3.92 non-reference field:** A *field* coded with *nal_ref_idc* equal to 0.
- 3.93 non-reference frame:** A *frame* coded with *nal_ref_idc* equal to 0.
- 3.94 non-reference picture:** A *picture* coded with *nal_ref_idc* equal to 0. A *non-reference picture* is not used for *inter prediction* of any other *pictures*.
- 3.95 note:** A term used to prefix *informative* remarks. This term is used exclusively in an *informative* context.
- 3.96 opposite parity:** The *opposite parity* of *top* is *bottom*, and vice versa.
- 3.97 output order:** The order in which the *decoded pictures* are output from the *decoded picture buffer*.
- 3.98 P slice:** A *slice* that may be decoded using *intra prediction* from decoded samples within the same *slice* or *inter prediction* from previously-decoded *reference pictures*, using at most one *motion vector* and *reference index* to *predict* the sample values of each *block*.
- 3.99 parameter:** A *syntax element* of a *sequence parameter set* or a *picture parameter set*. Parameter is also used as part of the defined term *quantisation parameter*.
- 3.100 parity:** The parity of a *field* can be *top* or *bottom*.
- 3.101 partitioning:** The division of a set into subsets such that each element of the set is in exactly one of the subsets.
- 3.102 picture:** A collective term for a *field* or a *frame*.
- 3.103 picture parameter set:** A *syntax structure* containing *syntax elements* that apply to zero or more entire *coded pictures* as determined by the *pic_parameter_set_id syntax element* found in each *slice header*.
- 3.104 picture order count:** A variable having a value that is non-decreasing with increasing *picture* position in output order relative to the previous *IDR picture* in *decoding order* or relative to the previous *picture* containing the *memory management control operation* that marks all *reference pictures* as “unused for reference”.
- 3.105 prediction:** An embodiment of the *prediction process*.
- 3.106 prediction process:** The use of a *predictor* to provide an estimate of the sample value or data element currently being decoded.
- 3.107 predictive slice:** See *P slice*.
- 3.108 predictor:** A combination of specified values or previously decoded sample values or data elements used in the *decoding process* of subsequent sample values or data elements.
- 3.109 primary coded picture:** The coded representation of a *picture* to be used by the *decoding process* for a bitstream conforming to this Recommendation | International Standard. The primary coded picture contains all *macroblocks* of the *picture*. The only *pictures* that have a normative effect on the *decoding process* are primary coded pictures. See also *redundant coded picture*.
- 3.110 profile:** A specified subset of the syntax of this Recommendation | International Standard.
- 3.111 quantisation parameter:** A variable used by the *decoding process* for *scaling* of *transform coefficient levels*.
- 3.112 random access:** The act of starting the decoding process for a *bitstream* at a point other than the beginning of the stream.
- 3.113 raster scan:** A mapping of a rectangular two-dimensional pattern to a one-dimensional pattern such that the first entries in the one-dimensional pattern are from the first top row of the two-dimensional pattern scanned from left to right, followed similarly by the second, third, etc. rows of the pattern (going down) each scanned from left to right.
- 3.114 raw byte sequence payload (RBSP):** A *syntax structure* containing an integer number of *bytes* that is encapsulated in a *NAL unit*. An RBSP is either empty or has the form of a *string of data bits* containing *syntax elements* followed by an *RBSP stop bit* and followed by zero or more subsequent bits equal to 0.
- 3.115 raw byte sequence payload (RBSP) stop bit:** A bit equal to 1 present within a *raw byte sequence payload (RBSP)* after a *string of data bits*. The location of the end of the *string of data bits* within an RBSP can be identified by searching from the end of the RBSP for the *RBSP stop bit*, which is the last non-zero bit in the RBSP.

- 3.116 recovery point:** A point in the *bitstream* at which the recovery of an exact or an approximate representation of the *decoded pictures* represented by the *bitstream* is achieved after a *random access* or *broken link*.
- 3.117 redundant coded picture:** A coded representation of a *picture* or a part of a *picture*. The content of a redundant coded picture shall not be used by the *decoding process* for a *bitstream* conforming to this Recommendation | International Standard. A *redundant coded picture* is not required to contain all *macroblocks* in the *primary coded picture*. Redundant coded pictures have no normative effect on the *decoding process*. See also *primary coded picture*.
- 3.118 reference field:** A *reference field* may be used for *inter prediction* when *P*, *SP*, and *B slices* of a *coded field* or *field macroblocks* of a *coded frame* are decoded. See also *reference picture*.
- 3.119 reference frame:** A *reference frame* may be used for *inter prediction* when *P*, *SP*, and *B slices* of a *coded frame* are decoded. See also *reference picture*.
- 3.120 reference index:** An index into a *reference picture list*.
- 3.121 reference picture:** A *picture* with *nal_ref_idc* not equal to 0. A *reference picture* contains samples that may be used for *inter prediction* in the *decoding process* of subsequent *pictures* in *decoding order*.
- 3.122 reference picture list:** A list of *reference pictures* that is used for *inter prediction* of a *P*, *B*, or *SP slice*. For the *decoding process* of a *P* or *SP slice*, there is one reference picture list. For the *decoding process* of a *B slice*, there are two reference picture lists.
- 3.123 reference picture list 0:** A *reference picture list* used for *inter prediction* of a *P*, *B*, or *SP slice*. All *inter prediction* used for *P* and *SP slices* uses reference picture list 0. Reference picture list 0 is one of two *reference picture lists* used for *inter prediction* for a *B slice*, with the other being *reference picture list 1*.
- 3.124 reference picture list 1:** A *reference picture list* used for *inter prediction* of a *B slice*. Reference picture list 1 is one of two lists of *reference picture lists* used for *inter prediction* for a *B slice*, with the other being *reference picture list 0*.
- 3.125 reference picture marking:** Specifies, in the *bitstream*, how the *decoded pictures* are marked for *inter prediction*.
- 3.126 reserved:** The term reserved, when used in the clauses specifying some values of a particular *syntax element*, are for future use by ITU-T | ISO/IEC. These values shall not be used in *bitstreams* conforming to this Recommendation | International Standard, but may be used in future extensions of this Recommendation | International Standard by ITU-T | ISO/IEC.
- 3.127 residual:** The decoded difference between a *prediction* of a sample or data element and its decoded value.
- 3.128 run:** A number of consecutive data elements represented in the decoding process. In one context, the number of zero-valued *transform coefficient levels* preceding a non-zero *transform coefficient level* in the list of *transform coefficient levels* generated by a *zig-zag scan* or a *field scan*. In other contexts, run refers to a number of *macroblocks*.
- 3.129 sample aspect ratio:** Specifies, for assisting the display process, which is not specified in this Recommendation | International Standard, the ratio between the intended horizontal distance between the columns and the intended vertical distance between the rows of the *luma* sample array in a *frame*. Sample aspect ratio is expressed as $h:v$, where h is horizontal width and v is vertical height (in arbitrary units of spatial distance).
- 3.130 scaling:** The process of multiplying *transform coefficient levels* by a factor, resulting in *transform coefficients*.
- 3.131 sequence parameter set:** A *syntax structure* containing *syntax elements* that apply to zero or more entire *coded video sequences* as determined by the content of a *seq_parameter_set_id syntax element* found in the *picture parameter set* referred to by the *pic_parameter_set_id syntax element* found in each *slice header*.
- 3.132 shall:** A term used to express mandatory requirements for conformance to this Recommendation | International Standard. When used to express a mandatory constraint on the values of *syntax elements* or on the results obtained by operation of the specified *decoding process*, it is the responsibility of the *encoder* to ensure that the constraint is fulfilled. When used in reference to operations performed by the *decoding process*, any *decoding process* that produces identical results to the *decoding process* described herein conforms to the *decoding process* requirements of this Recommendation | International Standard.
- 3.133 should:** A term used to refer to behaviour of an implementation that is encouraged to be followed under anticipated ordinary circumstances, but is not a mandatory requirement for conformance to this Recommendation | International Standard.

- 3.134 SI slice:** A *slice* that is coded using *prediction* only from decoded samples within the same *slice* and using quantisation of the *prediction* samples. An SI slice can be coded such that its decoded samples can be constructed identically to an *SP slice*.
- 3.135 skipped macroblock:** A *macroblock* for which no data is coded other than an indication that the *macroblock* is to be decoded as "skipped". This indication may be common to several *macroblocks*.
- 3.136 slice:** An integer number of *macroblocks* or *macroblock pairs* ordered consecutively in the *raster scan* within a particular *slice group*. For the *primary coded picture*, the division of each *slice group* into slices is a *partitioning*. Although a slice contains *macroblocks* or *macroblock pairs* that are consecutive in the *raster scan* within a *slice group*, these *macroblocks* or *macroblock pairs* are not necessarily consecutive in the *raster scan* within the *picture*. The addresses of the *macroblocks* are derived from the address of the first *macroblock* in a slice (as represented in the *slice header*) and the *macroblock to slice group map*.
- 3.137 slice data partitioning:** A method of *partitioning* selected *syntax elements* into *syntax structures* based on a *category* associated with each *syntax element*.
- 3.138 slice group:** A subset of the *macroblocks* or *macroblock pairs* of a *picture*. The division of the *picture* into slice groups is a *partitioning* of the *picture*. The partitioning is specified by the *macroblock to slice group map*.
- 3.139 slice group map units:** The units of the *map unit to slice group map*.
- 3.140 slice header:** A part of a coded *slice* containing the data elements pertaining to the first or all *macroblocks* represented in the *slice*.
- 3.141 source:** Term used to describe the video material or some of its attributes before encoding.
- 3.142 SP slice:** A *slice* that is coded using *inter prediction* from previously-decoded *reference pictures*, using at most one *motion vector* and *reference index* to *predict* the sample values of each *block*. An SP slice can be coded such that its decoded samples can be constructed identically to another SP slice or an *SI slice*.
- 3.143 start code prefix:** A unique sequence of three *bytes* equal to 0x000001 embedded in the *byte stream* as a prefix to each *NAL unit*. The location of a start code prefix can be used by a *decoder* to identify the beginning of a new *NAL unit* and the end of a previous *NAL unit*. Emulation of start code prefixes is prevented within *NAL units* by the inclusion of *emulation prevention bytes*.
- 3.144 string of data bits (SODB):** A sequence of some number of bits representing *syntax elements* present within a *raw byte sequence payload* prior to the *raw byte sequence payload stop bit*. Within an SODB, the left-most bit is considered to be the first and most significant bit, and the right-most bit is considered to be the last and least significant bit.
- 3.145 sub-macroblock:** One quarter of the samples of a *macroblock*, i.e., an 8x8 *luma block* and two corresponding *chroma blocks* of which one corner is located at a corner of the *macroblock*.
- 3.146 sub-macroblock partition:** A *block* of *luma* samples and two corresponding *blocks* of *chroma* samples resulting from a *partitioning* of a *sub-macroblock* for *inter prediction*.
- 3.147 switching I slice:** See *SI slice*.
- 3.148 switching P slice:** See *SP slice*.
- 3.149 syntax element:** An element of data represented in the *bitstream*.
- 3.150 syntax structure:** Zero or more *syntax elements* present together in the *bitstream* in a specified order.
- 3.151 top field:** One of two *fields* that comprise a *frame*. Each row of a *top field* is spatially located immediately above the corresponding row of the *bottom field*.
- 3.152 top macroblock (of a macroblock pair):** The *macroblock* within a *macroblock pair* that contains the samples in the top row of samples for the *macroblock pair*. For a *field macroblock pair*, the top macroblock represents the samples from the region of the *top field* of the *frame* that lie within the spatial region of the *macroblock pair*. For a *frame macroblock pair*, the top macroblock represents the samples of the *frame* that lie within the top half of the spatial region of the *macroblock pair*.
- 3.153 transform coefficient:** A scalar quantity, considered to be in a frequency domain, that is associated with a particular one-dimensional or two-dimensional *frequency index* in an *inverse transform* part of the *decoding process*.

- 3.154 transform coefficient level:** An integer quantity representing the value associated with a particular two-dimensional frequency index in the *decoding process* prior to *scaling* for computation of a *transform coefficient* value.
- 3.155 universal unique identifier (UUID):** An identifier that is unique with respect to the space of all universal unique identifiers.
- 3.156 unspecified:** The term unspecified, when used in the clauses specifying some values of a particular *syntax element*, indicates that the values have no specified meaning in this Recommendation | International Standard and will not have a specified meaning in the future as an integral part of this Recommendation | International Standard.
- 3.157 variable length coding (VLC):** A reversible procedure for entropy coding that assigns shorter bit strings to *symbols* expected to be more frequent and longer bit strings to *symbols* expected to be less frequent.
- 3.158 zig-zag scan:** A specific sequential ordering of *transform coefficient levels* from (approximately) the lowest spatial frequency to the highest. Zig-zag scan is used for *transform coefficient levels* in *frame macroblocks*.

4 Abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviations apply.

CABAC	Context-based Adaptive Binary Arithmetic Coding
CAVLC	Context-based Adaptive Variable Length Coding
CBR	Constant Bit Rate
CPB	Coded Picture Buffer
DPB	Decoded Picture Buffer
DUT	Decoder under test
FIFO	First-In, First-Out
HRD	Hypothetical Reference Decoder
HSS	Hypothetical Stream Scheduler
IDR	Instantaneous Decoding Refresh
LSB	Least Significant Bit
MB	Macroblock
MBAFF	Macroblock-Adaptive Frame-Field Coding
MSB	Most Significant Bit
NAL	Network Abstraction Layer
RBSP	Raw Byte Sequence Payload
SEI	Supplemental Enhancement Information
SODB	String Of Data Bits
UUID	Universal Unique Identifier
VBR	Variable Bit Rate
VCL	Video Coding Layer
VLC	Variable Length Coding
VUI	Video Usability Information

5 Conventions

NOTE – The mathematical operators used in this Specification are similar to those used in the C programming language. However, integer division and arithmetic shift operations are specifically defined. Numbering and counting conventions generally begin from 0.

5.1 Arithmetic operators

The following arithmetic operators are defined as follows.

+	Addition
−	Subtraction (as a two-argument operator) or negation (as a unary prefix operator)
*	Multiplication
x^y	Exponentiation. Specifies x to the power of y . In other contexts, such notation is used for superscripting not intended for interpretation as exponentiation.
/	Integer division with truncation of the result toward zero. For example, $7/4$ and $-7/-4$ are truncated to 1 and $-7/4$ and $7/-4$ are truncated to -1 .
÷	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\frac{x}{y}$	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\sum_{i=x}^y f(i)$	The summation of $f(i)$ with i taking all integer values from x up to and including y .
$x \% y$	Modulus. Remainder of x divided by y , defined only for integers x and y with $x \geq 0$ and $y > 0$.

When order of precedence is not indicated explicitly by use of parenthesis, the following rules apply:

- multiplication and division operations are considered to take place before addition and subtraction;
- multiplication and division operations in sequence are evaluated sequentially from left to right;
- addition and subtraction operations in sequence are evaluated sequentially from left to right.

5.2 Logical operators

The following logical operators are defined as follows:

$x \ \&\& \ y$	Boolean logical "and" of x and y
$x \ \ y$	Boolean logical "or" of x and y
!	Boolean logical "not"
$x \ ? \ y : z$	If x is TRUE or not equal to 0, evaluates to the value of y ; otherwise, evaluates to the value of z

5.3 Relational operators

The following relational operators are defined as follows:

>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
=	Equal to
!=	Not equal to

When a relational operator is applied to a syntax element or variable that has been assigned the value "na" (not applicable), the value "na" is treated as a distinct value for the syntax element or variable. The value "na" is considered not to be equal to any other value.

5.4 Bit-wise operators

The following bit-wise operators are defined as follows:

&	Bit-wise "and". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.
---	---

	Bit-wise "or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.
$x \gg y$	Arithmetic right shift of a two's complement integer representation of x by y binary digits. This function is defined only for positive integer values of y . Bits shifted into the MSBs as a result of the right shift have a value equal to the MSB of x prior to the shift operation.
$x \ll y$	Arithmetic left shift of a two's complement integer representation of x by y binary digits. This function is defined only for positive integer values of y . Bits shifted into the LSBs as a result of the left shift have a value equal to 0.

5.5 Assignment operators

The following arithmetic operators are defined as follows:

=	Assignment operator.
++	Increment, i.e., $x++$ is equivalent to $x = x + 1$; when used in an array index, evaluates to the value of the variable prior to the increment operation.
--	Decrement, i.e., $x--$ is equivalent to $x = x - 1$; when used in an array index, evaluates to the value of the variable prior to the decrement operation.
+=	Increment by amount specified, i.e., $x += 3$ is equivalent to $x = x + 3$, and $x += (-3)$ is equivalent to $x = x + (-3)$.
-=	Decrement by amount specified, i.e., $x -= 3$ is equivalent to $x = x - 3$, and $x -= (-3)$ is equivalent to $x = x - (-3)$.

5.6 Range notation

The following notation is used to specify a range of values

$x = y .. z$ x takes on integer values starting from y to z inclusive, with x , y , and z being integer numbers.

5.7 Mathematical functions

The following mathematical functions are defined as follows:

$$\text{Abs}(x) = \begin{cases} x & ; \quad x \geq 0 \\ -x & ; \quad x < 0 \end{cases} \quad (5-1)$$

$$\text{Ceil}(x) \quad \text{the smallest integer greater than or equal to } x. \quad (5-2)$$

$$\text{Clip1}_Y(x) = \text{Clip3}(0, (1 \ll \text{BitDepth}_Y) - 1, x) \quad (5-3)$$

$$\text{Clip1}_C(x) = \text{Clip3}(0, (1 \ll \text{BitDepth}_C) - 1, x) \quad (5-4)$$

$$\text{Clip3}(x, y, z) = \begin{cases} x & ; \quad z < x \\ y & ; \quad z > y \\ z & ; \quad \text{otherwise} \end{cases} \quad (5-5)$$

$$\text{Floor}(x) \quad \text{the greatest integer less than or equal to } x. \quad (5-6)$$

$$\text{InverseRasterScan}(a, b, c, d, e) = \begin{cases} (a \% (d/b)) * b; & e = 0 \\ (a / (d/b)) * c; & e = 1 \end{cases} \quad (5-7)$$

Log2(x) returns the base-2 logarithm of x. (5-8)

Log10(x) returns the base-10 logarithm of x. (5-9)

Median(x, y, z) = x + y + z – Min(x, Min(y, z)) – Max(x, Max(y, z)) (5-10)

Min(x, y) = $\begin{cases} x & ; \quad x \leq y \\ y & ; \quad x > y \end{cases}$ (5-11)

Max(x, y) = $\begin{cases} x & ; \quad x \geq y \\ y & ; \quad x < y \end{cases}$ (5-12)

Round(x) = Sign(x) * Floor(Abs(x) + 0.5) (5-13)

Sign(x) = $\begin{cases} 1 & ; \quad x \geq 0 \\ -1 & ; \quad x < 0 \end{cases}$ (5-14)

Sqrt(x) = \sqrt{x} (5-15)

5.8 Variables, syntax elements, and tables

Syntax elements in the bitstream are represented in **bold** type. Each syntax element is described by its name (all lower case letters with underscore characters), its one or two syntax categories, and one or two descriptors for its method of coded representation. The decoding process behaves according to the value of the syntax element and to the values of previously decoded syntax elements. When a value of a syntax element is used in the syntax tables or the text, it appears in regular (i.e., not bold) type.

In some cases the syntax tables may use the values of other variables derived from syntax elements values. Such variables appear in the syntax tables, or text, named by a mixture of lower case and upper case letter and without any underscore characters. Variables starting with an upper case letter are derived for the decoding of the current syntax structure and all depending syntax structures. Variables starting with an upper case letter may be used in the decoding process for later syntax structures mentioning the originating syntax structure of the variable. Variables starting with a lower case letter are only used within the subclause in which they are derived.

In some cases, "mnemonic" names for syntax element values or variable values are used interchangeably with their numerical values. Sometimes "mnemonic" names are used without any associated numerical values. The association of values and names is specified in the text. The names are constructed from one or more groups of letters separated by an underscore character. Each group starts with an upper case letter and may contain more upper case letters.

NOTE – The syntax is described in a manner that closely follows the C-language syntactic constructs.

Functions are described by their names, which are constructed as syntax element names, with left and right round parentheses including zero or more variable names (for definition) or values (for usage), separated by commas (if more than one variable).

A one-dimensional array is referred to as a list. A two-dimensional array is referred to as a matrix. Arrays can either be syntax elements or variables. Subscripts or square parentheses are used for the indexing of arrays. In reference to a visual depiction of a matrix, the first subscript is used as a row (vertical) index and the second subscript is used as a column (horizontal) index. The indexing order is reversed when using square parentheses rather than subscripts for indexing. Thus, an element of a matrix s at horizontal position x and vertical position y may be denoted either as s[x, y] or as s_{yx}.

Binary notation is indicated by enclosing the string of bit values by single quote marks. For example, '01000001' represents an eight-bit string having only its second and its last bits equal to 1.

Hexadecimal notation, indicated by prefixing the hexadecimal number by "0x", may be used instead of binary notation when the number of bits is an integer multiple of 4. For example, 0x41 represents an eight-bit string having only its second and its last bits equal to 1.

Numerical values not enclosed in single quotes and not prefixed by "0x" are decimal values.

A value equal to 0 represents a FALSE condition in a test statement. The value TRUE is represented by any other value different than zero.

5.9 Text description of logical operations

In the text, a statement of logical operations as would be described in pseudo-code as

```
if( condition 0 )
    statement 0
else if ( condition 1 )
    statement 1
...
else /* informative remark on remaining condition */
    statement n
```

may be described in the following manner:

... as follows / ... the following applies.

- If condition 0, statement 0
- Otherwise, if condition 1, statement 1
- ...
- Otherwise (informative remark on remaining condition), statement n

Each "If...Otherwise, if...Otherwise, ..." statement in the text is introduced with "... as follows" or "... the following applies" immediately followed by "If ... ". The last condition of the "If...Otherwise, if...Otherwise, ..." is always an "Otherwise, ...". Interleaved "If...Otherwise, if...Otherwise, ..." statements can be identified by matching "... as follows" or "... the following applies" with the ending "Otherwise, ...".

In the text, a statement of logical operations as would be described in pseudo-code as

```
if( condition 0a && condition 0b )
    statement 0
else if ( condition 1a || condition 1b )
    statement 1
...
else
    statement n
```

may be described in the following manner:

... as follows / ... the following applies.

- If all of the following conditions are true, statement 0
 - condition 0a
 - condition 0b
- Otherwise, if any of the following conditions are true, statement 1
 - condition 1a
 - condition 1b
- ...
- Otherwise, statement n

In the text, a statement of logical operations as would be described in pseudo-code as

```
if( condition 0 )
    statement 0
if ( condition 1 )
    statement 1
```

may be described in the following manner:

```
When condition 0, statement 0
When condition 1, statement 1
```

5.10 Processes

Processes are used to describe the decoding of syntax elements. A process has a separate specification and invoking. All syntax elements and upper case variables that pertain to the current syntax structure and depending syntax structures are available in the process specification and invoking. A process specification may also have a lower case variable explicitly specified as the input. Each process specification has explicitly specified an output. The output is a variable that can either be an upper case variable or a lower case variable.

The assignment of variables is specified as follows.

- If invoking a process, variables are explicitly assigned to lower case input or output variables of the process specification in case these do not have the same name.
- Otherwise (when the variables at the invoking and specification have the same name), assignment is implied.

In the specification of a process, a specific macroblock may be referred to by the variable name having a value equal to the address of the specific macroblock.

6 Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships

6.1 Bitstream formats

This subclause specifies the relationship between the NAL unit stream and byte stream, either of which are referred to as the bitstream.

The bitstream can be in one of two formats: the NAL unit stream format or the byte stream format. The NAL unit stream format is conceptually the more "basic" type. It consists of a sequence of syntax structures called NAL units. This sequence is ordered in decoding order. There are constraints imposed on the decoding order (and contents) of the NAL units in the NAL unit stream.

The byte stream format can be constructed from the NAL unit stream format by ordering the NAL units in decoding order and prefixing each NAL unit with a start code prefix and zero or more zero-valued bytes to form a stream of bytes. The NAL unit stream format can be extracted from the byte stream format by searching for the location of the unique start code prefix pattern within this stream of bytes. Methods of framing the NAL units in a manner other than use of the byte stream format are outside the scope of this Recommendation | International Standard. The byte stream format is specified in Annex B.

6.2 Source, decoded, and output picture formats

This subclause specifies the relationship between source and decoded frames and fields that is given via the bitstream.

The video source that is represented by the bitstream is a sequence of either or both frames or fields (called collectively pictures) in decoding order.

The source and decoded pictures (frames or fields) are each comprised of one or more sample arrays:

- Luma (Y) only (monochrome), with or without an auxiliary array.
- Luma and two Chroma (YCbCr or YCgCo), with or without an auxiliary array.
- Green, Blue and Red (GBR, also known as RGB), with or without an auxiliary array.

- Arrays representing other unspecified monochrome or tri-stimulus colour samplings (for example, YZX, also known as XYZ), with or without an auxiliary array.

For convenience of notation and terminology in this specification, the variables and terms associated with these arrays are referred to as luma (or L or Y) and chroma, where the two chroma arrays are referred to as Cb and Cr; regardless of the actual colour representation method in use. The actual colour representation method in use can be indicated in syntax that is specified in Annex E. The (monochrome) auxiliary arrays, which may or may not be present as auxiliary pictures in a coded video sequence, are optional for decoding and can be used for such purposes as alpha blending.

The variables SubWidthC, and SubHeightC are specified in Table 6-1, depending on the chroma format sampling structure, which is specified through chroma_format_idc. An entry marked as "-" in Table 6-1 denotes an undefined value for SubWidthC or SubHeightC. Other values of chroma_format_idc, SubWidthC, and SubHeightC may be specified in the future by ITU-T | ISO/IEC.

Table 6-1 – SubWidthC, and SubHeightC values derived from chroma_format_idc

chroma_format_idc	Chroma Format	SubWidthC	SubHeightC
0	monochrome	–	–
1	4:2:0	2	2
2	4:2:2	2	1
3	4:4:4	1	1

In monochrome sampling there is only one sample array, which is nominally considered the luma array.

In 4:2:0 sampling, each of the two chroma arrays has half the height and half the width of the luma array.

In 4:2:2 sampling, each of the two chroma arrays has the same height and half the width of the luma array.

In 4:4:4 sampling, each of the two chroma arrays has the same height and width as the luma array.

The width and height of the luma sample arrays are each an integer multiple of 16. In bitstreams using 4:2:0 chroma sampling, the width and height of chroma sample arrays are each an integer multiple of 8. In bitstreams using 4:2:2 sampling, the width of the chroma sample arrays is an integer multiple of 8 and the height is an integer multiple of 16. The height of a luma array that is coded as two separate fields or in macroblock-adaptive frame-field coding (see below) is an integer multiple of 32. In bitstreams using 4:2:0 chroma sampling, the height of each chroma array that is coded as two separate fields or in macroblock-adaptive frame-field coding (see below) is an integer multiple of 16. The width or height of pictures output from the decoding process need not be an integer multiple of 16 and can be specified using a cropping rectangle.

The syntax for the luma and (when present) chroma arrays are ordered such when data for all three colour components is present, the data for the luma array is first, followed by any data for the Cb array, followed by any data for the Cr array, unless otherwise specified.

The width of fields coded referring to a specific sequence parameter set is the same as that of frames coded referring to the same sequence parameter set (see below). The height of fields coded referring to a specific sequence parameter set is half that of frames coded referring to the same sequence parameter set (see below).

The number of bits necessary for the representation of each of the samples in the luma and chroma arrays in a video sequence is in the range of 8 to 12, and the number of bits used in the luma array may differ from the number of bits used in the chroma arrays.

When the value of chroma_format_idc is equal to 1, the nominal vertical and horizontal relative locations of luma and chroma samples in frames are shown in Figure 6-1. Alternative chroma sample relative locations may be indicated in video usability information (see Annex E).

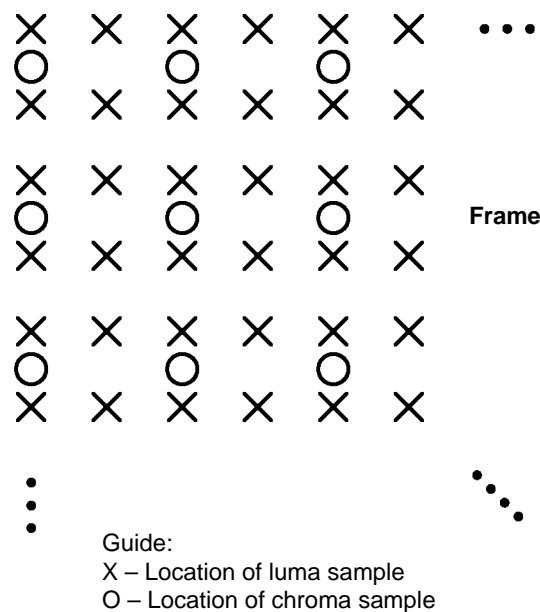


Figure 6-1 – Nominal vertical and horizontal locations of 4:2:0 luma and chroma samples in a frame

A frame consists of two fields as described below. A coded picture may represent a coded frame or an individual coded field. A coded video sequence conforming to this Recommendation | International Standard may contain arbitrary combinations of coded frames and coded fields. The decoding process is also specified in a manner that allows smaller regions of a coded frame to be coded either as a frame or field region, by use of macroblock-adaptive frame-field coding.

Source and decoded fields are one of two types: top field or bottom field. When two fields are output at the same time, or are combined to be used as a reference frame (see below), the two fields (which shall be of opposite parity) are interleaved. The first (i.e., top), third, fifth, etc. rows of a decoded frame are the top field rows. The second, fourth, sixth, etc. rows of a decoded frame are the bottom field rows. A top field consists of only the top field rows of a decoded frame. When the top field or bottom field of a decoded frame is used as a reference field (see below) only the even rows (for a top field) or the odd rows (for a bottom field) of the decoded frame are used.

When the value of `chroma_format_idc` is equal to 1, the nominal vertical and horizontal relative locations of luma and chroma samples in top and bottom fields are shown in Figure 6-2. The nominal vertical sampling relative locations of the chroma samples in a top field are specified as shifted up by one-quarter luma sample height relative to the field-sampling grid. The vertical sampling locations of the chroma samples in a bottom field are specified as shifted down by one-quarter luma sample height relative to the field-sampling grid. Alternative chroma sample relative locations may be indicated in the video usability information (see Annex E).

NOTE – The shifting of the chroma samples is in order for these samples to align vertically to the usual location relative to the full-frame sampling grid as shown in Figure 6-1.

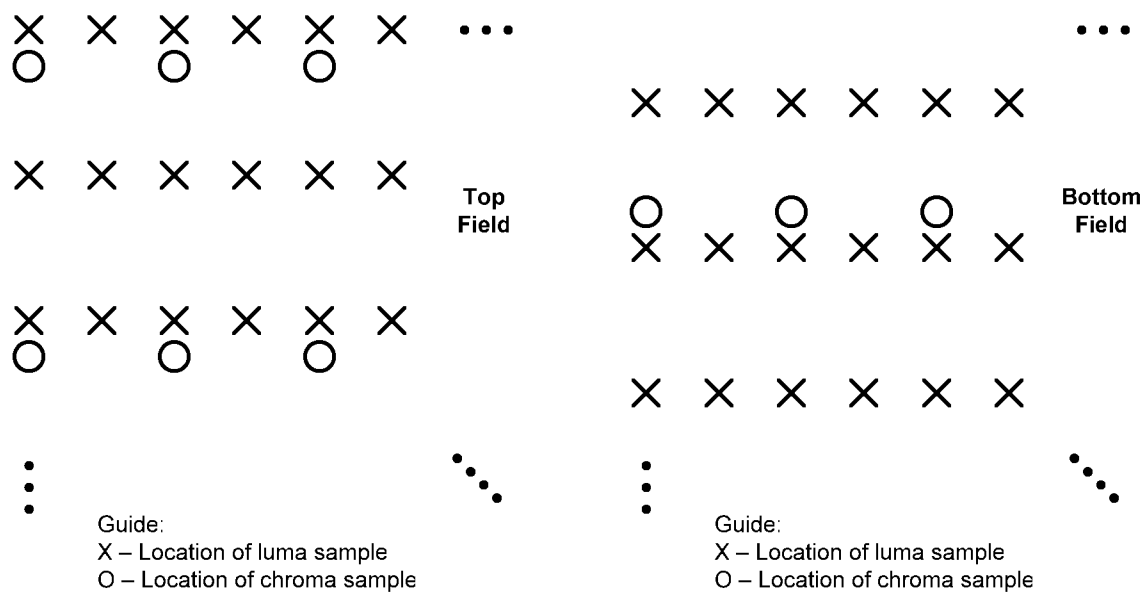


Figure 6-2 – Nominal vertical and horizontal sampling locations of 4:2:0 samples in top and bottom fields

When the value of `chroma_format_idc` is equal to 2, the chroma samples are co-sited with the corresponding luma samples and the nominal locations in a frame and in fields are as shown in Figure 6-3 – Nominal vertical and horizontal locations of 4:2:2 luma and chroma samples in a frame and Figure 6-4, respectively.

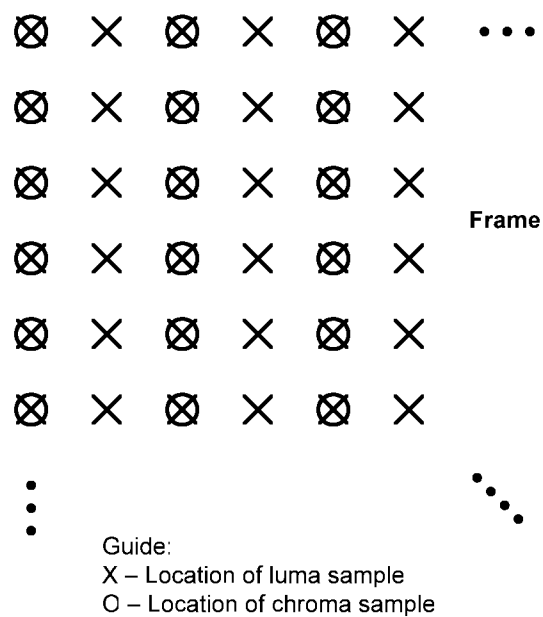


Figure 6-3 – Nominal vertical and horizontal locations of 4:2:2 luma and chroma samples in a frame

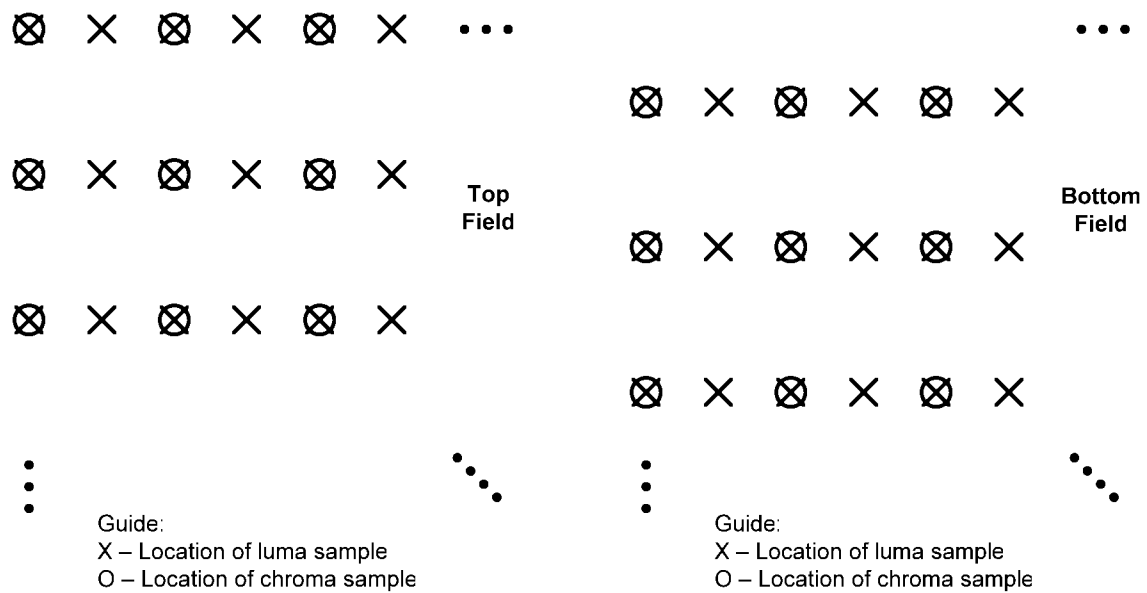


Figure 6-4 – Nominal vertical and horizontal sampling locations of 4:2:2 samples top and bottom fields

When the value of chroma_format_idc is equal to 3, all array samples are co-sited for all cases of frames and fields and the nominal locations in a frame and in fields are as shown in Figures 6-5 and 6-6, respectively.

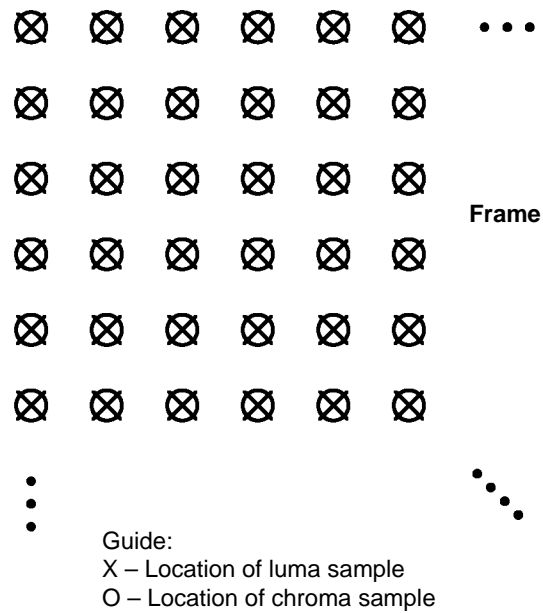


Figure 6-5 – Nominal vertical and horizontal locations of 4:4:4 luma and chroma samples in a frame

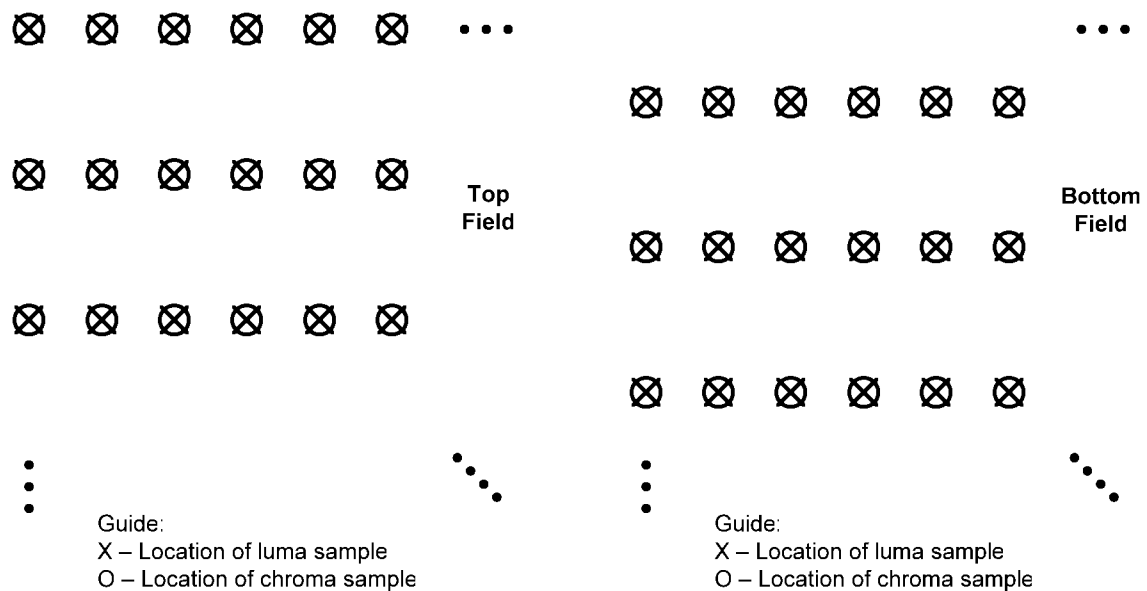


Figure 6-6 – Nominal vertical and horizontal sampling locations of 4:4:4 samples top and bottom fields

The samples are processed in units of macroblocks. The luma array for each macroblock is 16 samples in both width and height. The variables MbWidthC and MbHeightC, which specify the width and height, respectively, of the chroma arrays for each macroblock, are derived as follows.

- If chroma_format_idc is equal to 0 (monochrome), MbWidthC and MbHeightC are both equal to 0 (as no chroma arrays are specified for monochrome video).
- Otherwise, MbWidthC and MbHeightC are derived as

$$\text{MbWidthC} = 16 / \text{SubWidthC} \quad (6-1)$$

$$\text{MbHeightC} = 16 / \text{SubHeightC} \quad (6-2)$$

6.3 Spatial subdivision of pictures and slices

This subclause specifies how a picture is partitioned into slices and macroblocks. Pictures are divided into slices. A slice is a sequence of macroblocks, or, when macroblock-adaptive frame/field decoding is in use, a sequence of macroblock pairs.

Each macroblock is comprised of one 16x16 luma array and, when the video format is not monochrome, two corresponding chroma sample arrays. When macroblock-adaptive frame/field decoding is not in use, each macroblock represents a spatial rectangular region of the picture. For example, a picture may be divided into two slices as shown in Figure 6-7.

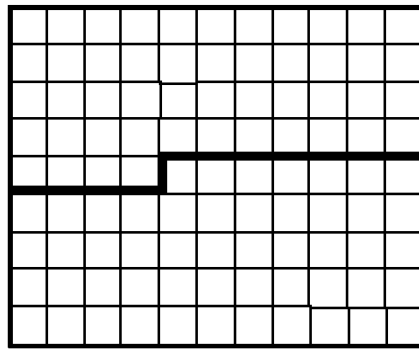


Figure 6-7 – A picture with 11 by 9 macroblocks that is partitioned into two slices

When macroblock-adaptive frame/field decoding is in use, the picture is partitioned into slices containing an integer number of macroblock pairs as shown in Figure 6-8. Each macroblock pair consists of two macroblocks.

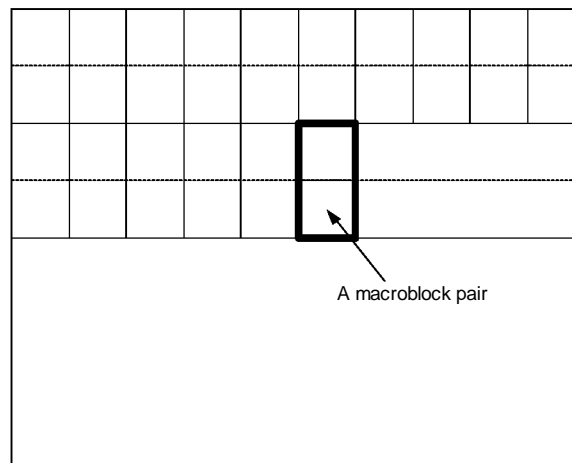


Figure 6-8 – Partitioning of the decoded frame into macroblock pairs

6.4 Inverse scanning processes and derivation processes for neighbours

This subclause specifies inverse scanning processes; i.e., the mapping of indices to locations, and derivation processes for neighbours.

6.4.1 Inverse macroblock scanning process

Input to this process is a macroblock address mbAddr.

Output of this process is the location (x, y) of the upper-left luma sample for the macroblock with address mbAddr relative to the upper-left sample of the picture.

The inverse macroblock scanning process is specified as follows.

- If MbaffFrameFlag is equal to 0,

$$x = \text{InverseRasterScan}(\text{mbAddr}, 16, 16, \text{PicWidthInSamples}_L, 0) \quad (6-3)$$

$$y = \text{InverseRasterScan}(\text{mbAddr}, 16, 16, \text{PicWidthInSamples}_L, 1) \quad (6-4)$$

- Otherwise (MbaffFrameFlag is equal to 1), the following applies.

$$xO = \text{InverseRasterScan}(\text{mbAddr} / 2, 16, 32, \text{PicWidthInSamples}_L, 0) \quad (6-5)$$

$$yO = \text{InverseRasterScan}(\text{mbAddr} / 2, 16, 32, \text{PicWidthInSamples}_L, 1) \quad (6-6)$$

Depending on the current macroblock the following applies.

- If the current macroblock is a frame macroblock

$$x = xO \quad (6-7)$$

$$y = yO + (\text{mbAddr} \% 2) * 16 \quad (6-8)$$

- Otherwise (the current macroblock is a field macroblock),

$$x = xO \quad (6-9)$$

$$y = yO + (\text{mbAddr} \% 2) \quad (6-10)$$

6.4.2 Inverse macroblock partition and sub-macroblock partition scanning process

Macroblocks or sub-macroblocks may be partitioned, and the partitions are scanned for inter prediction as shown in Figure 6-9. The outer rectangles refer to the samples in a macroblock or sub-macroblock, respectively. The rectangles refer to the partitions. The number in each rectangle specifies the index of the inverse macroblock partition scan or inverse sub-macroblock partition scan.

The functions MbPartWidth(), MbPartHeight(), SubMbPartWidth(), and SubMbPartHeight() describing the width and height of macroblock partitions and sub-macroblock partitions are specified in Tables 7-13, 7-14, 7-17, and 7-18. MbPartWidth() and MbPartHeight() are set to appropriate values for each macroblock, depending on the macroblock type. SubMbPartWidth() and SubMbPartHeight() are set to appropriate values for each sub-macroblock of a macroblock with mb_type equal to P_8x8, P_8x8ref0, or B_8x8, depending on the sub-macroblock type.

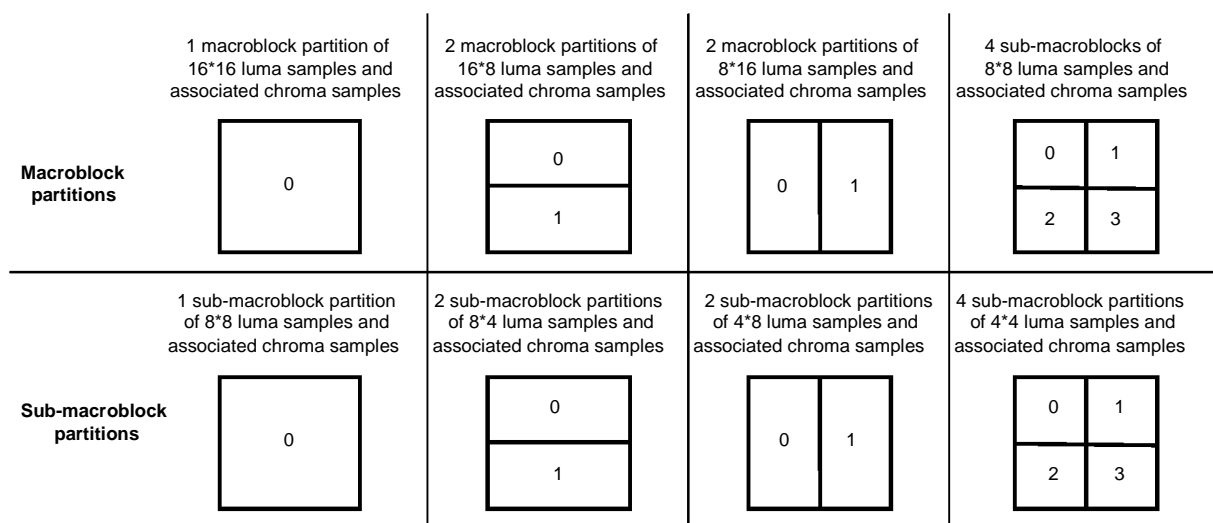


Figure 6-9 – Macroblock partitions, sub-macroblock partitions, macroblock partition scans, and sub-macroblock partition scans

6.4.2.1 Inverse macroblock partition scanning process

Input to this process is the index of a macroblock partition $mbPartIdx$.

Output of this process is the location (x, y) of the upper-left luma sample for the macroblock partition $mbPartIdx$ relative to the upper-left sample of the macroblock.

The inverse macroblock partition scanning process is specified by

$$x = \text{InverseRasterScan}(mbPartIdx, MbPartWidth(mb_type), MbPartHeight(mb_type), 16, 0) \quad (6-11)$$

$$y = \text{InverseRasterScan}(mbPartIdx, MbPartWidth(mb_type), MbPartHeight(mb_type), 16, 1) \quad (6-12)$$

6.4.2.2 Inverse sub-macroblock partition scanning process

Inputs to this process are the index of a macroblock partition $mbPartIdx$ and the index of a sub-macroblock partition $subMbPartIdx$.

Output of this process is the location (x, y) of the upper-left luma sample for the sub-macroblock partition $subMbPartIdx$ relative to the upper-left sample of the sub-macroblock.

The inverse sub-macroblock partition scanning process is specified as follows.

- If mb_type is equal to P_8x8, P_8x8ref0, or B_8x8,

$$x = \text{InverseRasterScan}(subMbPartIdx, \text{SubMbPartWidth}(sub_mb_type[mbPartIdx]), \text{SubMbPartHeight}(sub_mb_type[mbPartIdx]), 8, 0) \quad (6-13)$$

$$y = \text{InverseRasterScan}(subMbPartIdx, \text{SubMbPartWidth}(sub_mb_type[mbPartIdx]), \text{SubMbPartHeight}(sub_mb_type[mbPartIdx]), 8, 1) \quad (6-14)$$

- Otherwise,

$$x = \text{InverseRasterScan}(subMbPartIdx, 4, 4, 8, 0) \quad (6-15)$$

$$y = \text{InverseRasterScan}(subMbPartIdx, 4, 4, 8, 1) \quad (6-16)$$

6.4.3 Inverse 4x4 luma block scanning process

Input to this process is the index of a 4x4 luma block luma4x4BlkIdx.

Output of this process is the location (x, y) of the upper-left luma sample for the 4x4 luma block with index luma4x4BlkIdx relative to the upper-left luma sample of the macroblock.

Figure 6-10 shows the scan for the 4x4 luma blocks.

0	1	4	5
2	3	6	7
8	9	12	13
10	11	14	15

Figure 6-10 – Scan for 4x4 luma blocks

The inverse 4x4 luma block scanning process is specified by

$$x = \text{InverseRasterScan}(\text{luma4x4BlkIdx} / 4, 8, 8, 16, 0) + \text{InverseRasterScan}(\text{luma4x4BlkIdx} \% 4, 4, 4, 8, 0) \quad (6-17)$$

$$y = \text{InverseRasterScan}(\text{luma4x4BlkIdx} / 4, 8, 8, 16, 1) + \text{InverseRasterScan}(\text{luma4x4BlkIdx} \% 4, 4, 4, 8, 1) \quad (6-18)$$

6.4.4 Inverse 8x8 luma block scanning process

Input to this process is the index of an 8x8 luma block luma8x8BlkIdx.

Output of this process is the location (x, y) of the upper-left luma sample for the 8x8 luma block with index luma8x8BlkIdx relative to the upper-left luma sample of the macroblock.

Figure 6-11 shows the scan for the 8x8 luma blocks.

0	1
2	3

Figure 6-11 – Scan for 8x8 luma blocks

The inverse 8x8 luma block scanning process is specified by

$$x = \text{InverseRasterScan}(\text{luma8x8BlkIdx}, 8, 8, 16, 0) \quad (6-19)$$

$$y = \text{InverseRasterScan}(\text{luma8x8BlkIdx}, 8, 8, 16, 1) \quad (6-20)$$

6.4.5 Derivation process of the availability for macroblock addresses

Input to this process is a macroblock address mbAddr.

Output of this process is the availability of the macroblock mbAddr.

NOTE – The meaning of availability is determined when this process is invoked.

The macroblock is marked as available, unless one of the following conditions is true in which case the macroblock is marked as not available:

- $mbAddr < 0$
- $mbAddr > CurrMbAddr$
- the macroblock with address $mbAddr$ belongs to a different slice than the macroblock with address $CurrMbAddr$

6.4.6 Derivation process for neighbouring macroblock addresses and their availability

This process can only be invoked when $MbaffFrameFlag$ is equal to 0.

The outputs of this process are

- $mbAddrA$: the address and availability status of the macroblock to the left of the current macroblock.
- $mbAddrB$: the address and availability status of the macroblock above the current macroblock.
- $mbAddrC$: the address and availability status of the macroblock above-right of the current macroblock.
- $mbAddrD$: the address and availability status of the macroblock above-left of the current macroblock.

Figure 6-12 shows the relative spatial locations of the macroblocks with $mbAddrA$, $mbAddrB$, $mbAddrC$, and $mbAddrD$ relative to the current macroblock with $CurrMbAddr$.

$mbAddrD$	$mbAddrB$	$mbAddrC$
$mbAddrA$	$CurrMbAddr$	

Figure 6-12 – Neighbouring macroblocks for a given macroblock

Input to the process in subclause 6.4.5 is $mbAddrA = CurrMbAddr - 1$ and the output is whether the macroblock $mbAddrA$ is available. In addition, $mbAddrA$ is marked as not available when $CurrMbAddr \% PicWidthInMbs$ is equal to 0.

Input to the process in subclause 6.4.5 is $mbAddrB = CurrMbAddr - PicWidthInMbs$ and the output is whether the macroblock $mbAddrB$ is available.

Input to the process in subclause 6.4.5 is $mbAddrC = CurrMbAddr - PicWidthInMbs + 1$ and the output is whether the macroblock $mbAddrC$ is available. In addition, $mbAddrC$ is marked as not available when $(CurrMbAddr + 1) \% PicWidthInMbs$ is equal to 0.

Input to the process in subclause 6.4.5 is $mbAddrD = CurrMbAddr - PicWidthInMbs - 1$ and the output is whether the macroblock $mbAddrD$ is available. In addition, $mbAddrD$ is marked as not available when $CurrMbAddr \% PicWidthInMbs$ is equal to 0.

6.4.7 Derivation process for neighbouring macroblock addresses and their availability in MBAFF frames

This process can only be invoked when $MbaffFrameFlag$ is equal to 1.

The outputs of this process are

- $mbAddrA$: the address and availability status of the top macroblock of the macroblock pair to the left of the current macroblock pair.
- $mbAddrB$: the address and availability status of the top macroblock of the macroblock pair above the current macroblock pair.

- mbAddrC: the address and availability status of the top macroblock of the macroblock pair above-right of the current macroblock pair.
- mbAddrD: the address and availability status of the top macroblock of the macroblock pair above-left of the current macroblock pair.

Figure 6-13 shows the relative spatial locations of the macroblocks with mbAddrA, mbAddrB, mbAddrC, and mbAddrD relative to the current macroblock with CurrMbAddr.

mbAddrA, mbAddrB, mbAddrC, and mbAddrD have identical values regardless whether the current macroblock is the top or the bottom macroblock of a macroblock pair.

mbAddrD	mbAddrB	mbAddrC
mbAddrA	CurrMbAddr or CurrMbAddr	

Figure 6-13 – Neighbouring macroblocks for a given macroblock in MBAFF frames

Input to the process in subclause 6.4.5 is $mbAddrA = 2 * (CurrMbAddr / 2 - 1)$ and the output is whether the macroblock mbAddrA is available. In addition, mbAddrA is marked as not available when $(CurrMbAddr / 2) \% PicWidthInMbs$ is equal to 0.

Input to the process in subclause 6.4.5 is $mbAddrB = 2 * (CurrMbAddr / 2 - PicWidthInMbs)$ and the output is whether the macroblock mbAddrB is available.

Input to the process in subclause 6.4.5 is $mbAddrC = 2 * (CurrMbAddr / 2 - PicWidthInMbs + 1)$ and the output is whether the macroblock mbAddrC is available. In addition, mbAddrC is marked as not available when $(CurrMbAddr / 2 + 1) \% PicWidthInMbs$ is equal to 0.

Input to the process in subclause 6.4.5 is $mbAddrD = 2 * (CurrMbAddr / 2 - PicWidthInMbs - 1)$ and the output is whether the macroblock mbAddrD is available. In addition, mbAddrD is marked as not available when $(CurrMbAddr / 2) \% PicWidthInMbs$ is equal to 0.

6.4.8 Derivation processes for neighbouring macroblocks, blocks, and partitions

Subclause 6.4.8.1 specifies the derivation process for neighbouring macroblocks.

Subclause 6.4.8.2 specifies the derivation process for neighbouring 8x8 luma blocks.

Subclause 6.4.8.3 specifies the derivation process for neighbouring 4x4 luma blocks.

Subclause 6.4.8.4 specifies the derivation process for neighbouring 4x4 chroma blocks.

Subclause 6.4.8.5 specifies the derivation process for neighbouring partitions.

Table 6-2 specifies the values for the difference of luma location (xD, yD) for the input and the replacement for N in mbAddrN, mbPartIdxN, subMbPartIdxN, luma8x8BlkIdxN, luma4x4BlkIdxN, and chroma4x4BlkIdxN for the output. These input and output assignments are used in subclauses 6.4.8.1 to 6.4.8.5. The variable predPartWidth is specified when Table 6-2 is referred to.

Table 6-2 – Specification of input and output assignments for subclauses 6.4.8.1 to 6.4.8.5

N	xD	yD
A	-1	0
B	0	-1
C	predPartWidth	-1
D	-1	-1

Figure 6-14 illustrates the relative location of the neighbouring macroblocks, blocks, or partitions A, B, C, and D to the current macroblock, partition, or block, when the current macroblock, partition, or block is in frame coding mode.

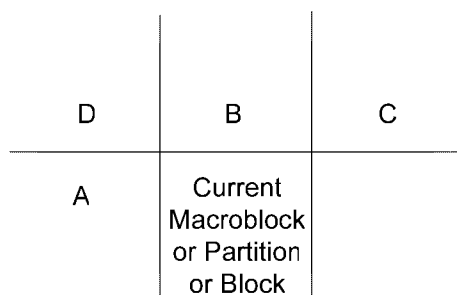


Figure 6-14 – Determination of the neighbouring macroblock, blocks, and partitions (informative)

6.4.8.1 Derivation process for neighbouring macroblocks

Outputs of this process are

- mbAddrA: the address of the macroblock to the left of the current macroblock and its availability status and
- mbAddrB: the address of the macroblock above the current macroblock and its availability status.
- mbAddrN (with N being A or B) is derived as follows.
- The difference of luma location (xD, yD) is set according to Table 6-2.
- The derivation process for neighbouring locations as specified in subclause 6.4.9 is invoked for luma locations with (xN, yN) equal to (xD, yD), and the output is assigned to mbAddrN.

6.4.8.2 Derivation process for neighbouring 8x8 luma block

Input to this process is an 8x8 luma block index luma8x8BlkIdx.

The luma8x8BlkIdx specifies the 8x8 luma blocks of a macroblock in a raster scan.

Outputs of this process are

- mbAddrA: either equal to CurrMbAddr or the address of the macroblock to the left of the current macroblock and its availability status,
- luma8x8BlkIdxA: the index of the 8x8 luma block to the left of the 8x8 block with index luma8x8BlkIdx and its availability status,
- mbAddrB: either equal to CurrMbAddr or the address of the macroblock above the current macroblock and its availability status,
- luma8x8BlkIdxB: the index of the 8x8 luma block above the 8x8 block with index luma8x8BlkIdx and its availability status.

mbAddrN and luma8x8BlkIdxN (with N being A or B) are derived as follows.

- The difference of luma location (xD, yD) is set according to Table 6-2.
- The luma location (xN, yN) is specified by

$$xN = (\text{luma8x8BlkIdx} \% 2) * 8 + xD \quad (6-21)$$

$$yN = (\text{luma8x8BlkIdx} / 2) * 8 + yD \quad (6-22)$$

- The derivation process for neighbouring locations as specified in subclause 6.4.9 is invoked for luma locations with (xN, yN) as the input and the output is assigned to mbAddrN and (xW, yW).
- The variable luma8x8BlkIdxN is derived as follows.
 - If mbAddrN is not available, luma8x8BlkIdxN is marked as not available.
 - Otherwise (mbAddrN is available), the 8x8 luma block in the macroblock mbAddrN covering the luma location (xW, yW) is assigned to luma8x8BlkIdxN.

6.4.8.3 Derivation process for neighbouring 4x4 luma blocks

Input to this process is a 4x4 luma block index luma4x4BlkIdx.

Outputs of this process are

- mbAddrA: either equal to CurrMbAddr or the address of the macroblock to the left of the current macroblock and its availability status,
- luma4x4BlkIdxA: the index of the 4x4 luma block to the left of the 4x4 block with index luma4x4BlkIdx and its availability status,
- mbAddrB: either equal to CurrMbAddr or the address of the macroblock above the current macroblock and its availability status,
- luma4x4BlkIdxB: the index of the 4x4 luma block above the 4x4 block with index luma4x4BlkIdx and its availability status.

mbAddrN and luma4x4BlkIdxN (with N being A or B) are derived as follows.

- The difference of luma location (xD, yD) is set according to Table 6-2.
- The inverse 4x4 luma block scanning process as specified in subclause 6.4.3 is invoked with luma4x4BlkIdx as the input and (x, y) as the output.
- The luma location (xN, yN) is specified by

$$xN = x + xD \quad (6-23)$$

$$yN = y + yD \quad (6-24)$$

- The derivation process for neighbouring locations as specified in subclause 6.4.9 is invoked for luma locations with (xN, yN) as the input and the output is assigned to mbAddrN and (xW, yW).
- The variable luma4x4BlkIdxN is derived as follows.
 - If mbAddrN is not available, luma4x4BlkIdxN is marked as not available.
 - Otherwise (mbAddrN is available), the 4x4 luma block in the macroblock mbAddrN covering the luma location (xW, yW) is assigned to luma4x4BlkIdxN.

6.4.8.4 Derivation process for neighbouring 4x4 chroma blocks

Input to this process is a 4x4 chroma block index chroma4x4BlkIdx.

Outputs of this process are

- mbAddrA (either equal to CurrMbAddr or the address of the macroblock to the left of the current macroblock) and its availability status,
- chroma4x4BlkIdxA (the index of the 4x4 chroma block to the left of the 4x4 chroma block with index chroma4x4BlkIdx) and its availability status,

- mbAddrB (either equal to CurrMbAddr or the address of the macroblock above the current macroblock) and its availability status,
- chroma4x4BlkIdxB (the index of the 4x4 chroma block above the 4x4 chroma block with index chroma4x4BlkIdx) and its availability status.

mbAddrN and chroma4x4BlkIdxN (with N being A or B) are derived as follows.

- The difference of chroma location (xD, yD) is set according to Table 6-2.
- Depending on chroma_format_idc, the position (x, y) of the upper-left sample of the 4x4 chroma block with index chroma4x4BlkIdx is derived as follows

- If chroma_format_idc is equal to 1 or 2, the following applies

$$x = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 0) \quad (6-25)$$

$$y = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 1) \quad (6-26)$$

- Otherwise (chroma_format_idc is equal to 3), the following applies

$$x = \text{InverseRasterScan}(\text{chroma4x4BlkIdx} / 4, 8, 8, 16, 0) + \text{InverseRasterScan}(\text{chroma4x4BlkIdx} \% 4, 4, 4, 8, 0) \quad (6-27)$$

$$y = \text{InverseRasterScan}(\text{chroma4x4BlkIdx} / 4, 8, 8, 16, 1) + \text{InverseRasterScan}(\text{chroma4x4BlkIdx} \% 4, 4, 4, 8, 1) \quad (6-28)$$

- The chroma location (xN, yN) is specified by

$$xN = x + xD \quad (6-29)$$

$$yN = y + yD \quad (6-30)$$

- The derivation process for neighbouring locations as specified in subclause 6.4.9 is invoked for chroma locations with (xN, yN) as the input and the output is assigned to mbAddrN and (xW, yW).
- The variable chroma4x4BlkIdxN is derived as follows.
 - If mbAddrN is not available, chroma4x4BlkIdxN is marked as not available.
 - Otherwise (mbAddrN is available), the 4x4 chroma block in the macroblock mbAddrN covering the chroma location (xW, yW) is assigned to chroma4x4BlkIdxN.

6.4.8.5 Derivation process for neighbouring partitions

Inputs to this process are

- a macroblock partition index mbPartIdx
- a current sub-macroblock type currSubMbType
- a sub-macroblock partition index subMbPartIdx

Outputs of this process are

- mbAddrA\mbPartIdxA\subMbPartIdxA: specifying the macroblock or sub-macroblock partition to the left of the current macroblock and its availability status, or the sub-macroblock partition CurrMbAddr\mbPartIdx\subMbPartIdx and its availability status,

- mbAddrB\mbPartIdxB\subMbPartIdxB: specifying the macroblock or sub-macroblock partition above the current macroblock and its availability status, or the sub-macroblock partition CurrMbAddr\mbPartIdx\subMbPartIdx and its availability status,
- mbAddrC\mbPartIdxC\subMbPartIdxC: specifying the macroblock or sub-macroblock partition to the right-above of the current macroblock and its availability status, or the sub-macroblock partition CurrMbAddr\mbPartIdx\subMbPartIdx and its availability status,
- mbAddrD\mbPartIdxD\subMbPartIdxD: specifying the macroblock or sub-macroblock partition to the left-above of the current macroblock and its availability status, or the sub-macroblock partition CurrMbAddr\mbPartIdx\subMbPartIdx and its availability status.

mbAddrN, mbPartIdxN, and subMbPartIdx (with N being A, B, C, or D) are derived as follows.

- The inverse macroblock partition scanning process as described in subclause 6.4.2.1 is invoked with mbPartIdx as the input and (x, y) as the output.
- The location of the upper-left luma sample inside a macroblock partition (xS, yS) is derived as follows.
 - If mb_type is equal to P_8x8, P_8x8ref0 or B_8x8, the inverse sub-macroblock partition scanning process as described in subclause 6.4.2.2 is invoked with subMbPartIdx as the input and (xS, yS) as the output.
 - Otherwise, (xS, yS) are set to (0, 0).
- The variable predPartWidth in Table 6-2 is specified as follows.
 - If mb_type is equal to P_Skip, B_Skip, or B_Direct_16x16, predPartWidth = 16.
 - Otherwise, if mb_type is equal to B_8x8, the following applies.
 - If currSubMbType is equal to B_Direct_8x8, predPartWidth = 16.
NOTE 1 – When currSubMbType is equal to B_Direct_8x8 and direct_spatial_mv_pred_flag is equal to 1, the predicted motion vector is the predicted motion vector for the complete macroblock.
 - Otherwise, predPartWidth = SubMbPartWidth(sub_mb_type[mbPartIdx]).
 - Otherwise, if mb_type is equal to P_8x8 or P_8x8ref0, predPartWidth = SubMbPartWidth(sub_mb_type[mbPartIdx]).
 - Otherwise, predPartWidth = MbPartWidth(mb_type).
- The difference of luma location (xD, yD) is set according to Table 6-2.
- The neighbouring luma location (xN, yN) is specified by

$$xN = x + xS + xD \quad (6-31)$$

$$yN = y + yS + yD \quad (6-32)$$

- The derivation process for neighbouring locations as specified in subclause 6.4.9 is invoked for luma locations with (xN, yN) as the input and the output is assigned to mbAddrN and (xW, yW).
- Depending on mbAddrN, the following applies.
 - If mbAddrN is not available, the macroblock or sub-macroblock partition mbAddrN\mbPartIdxN\subMbPartIdxN is marked as not available.
 - Otherwise (mbAddrN is available), the following applies.
 - The macroblock partition in the macroblock mbAddrN covering the luma location (xW, yW) is assigned to mbPartIdxN and the sub-macroblock partition inside the macroblock partition mbPartIdxN covering the sample (xW, yW) in the macroblock mbAddrN is assigned to subMbPartIdxN.
 - When the partition given by mbPartIdxN and subMbPartIdxN is not yet decoded, the macroblock partition mbPartIdxN and the sub-macroblock partition subMbPartIdxN are marked as not available.

NOTE 2 – The latter condition is, for example, the case when mbPartIdx = 2, subMbPartIdx = 3, xD = 4, yD = -1, i.e., when neighbour C of the last 4x4 luma block of the third sub-macroblock is requested.

6.4.9 Derivation process for neighbouring locations

Input to this process is a luma or chroma location (x_N, y_N) expressed relative to the upper left corner of the current macroblock.

Outputs of this process are

- $mbAddrN$: either equal to $CurrMbAddr$ or to the address of neighbouring macroblock that contains (x_N, y_N) and its availability status,
- (x_W, y_W): the location (x_N, y_N) expressed relative to the upper-left corner of the macroblock $mbAddrN$ (rather than relative to the upper-left corner of the current macroblock).

Let $maxW$ and $maxH$ be variables specifying maximum values of the location components x_N, x_W , and y_N, y_W , respectively. $maxW$ and $maxH$ are derived as follows.

- If this process is invoked for neighbouring luma locations,

$$maxW = maxH = 16 \quad (6-33)$$

- Otherwise (this process is invoked for neighbouring chroma locations),

$$maxW = MbWidthC \quad (6-34)$$

$$maxH = MbHeightC \quad (6-35)$$

Depending on the variable $MbaffFrameFlag$, the neighbouring locations are derived as follows.

- If $MbaffFrameFlag$ is equal to 0, the specification for neighbouring locations in fields and non-MBAFF frames as described in subclause 6.4.9.1 is applied.
- Otherwise ($MbaffFrameFlag$ is equal to 1), the specification for neighbouring locations in MBAFF frames as described in subclause 6.4.9.2 is applied.

6.4.9.1 Specification for neighbouring locations in fields and non-MBAFF frames

The specifications in this subclause are applied when $MbaffFrameFlag$ is equal to 0.

The derivation process for neighbouring macroblock addresses and their availability in subclause 6.4.6 is invoked with $mbAddrA, mbAddrB, mbAddrC$, and $mbAddrD$ as well as their availability status as the output.

Table 6-3 specifies $mbAddrN$ depending on (x_N, y_N).

Table 6-3 – Specification of $mbAddrN$

x_N	y_N	$mbAddrN$
< 0	< 0	$mbAddrD$
< 0	$0 .. maxH - 1$	$mbAddrA$
$0 .. maxW - 1$	< 0	$mbAddrB$
$0 .. maxW - 1$	$0 .. maxH - 1$	$CurrMbAddr$
$> maxW - 1$	< 0	$mbAddrC$
$> maxW - 1$	$0 .. maxH - 1$	not available
	$> maxH - 1$	not available

The neighbouring location (x_W, y_W) relative to the upper-left corner of the macroblock $mbAddrN$ is derived as

$$x_W = (x_N + maxW) \% maxW \quad (6-36)$$

$$y_W = (y_N + maxH) \% maxH \quad (6-37)$$

6.4.9.2 Specification for neighbouring locations in MBAFF frames

The specifications in this subclause are applied when MbaffFrameFlag is equal to 1.

The derivation process for neighbouring macroblock addresses and their availability in subclause 6.4.7 is invoked with mbAddrA, mbAddrB, mbAddrC, and mbAddrD as well as their availability status as the output.

Table 6-4 specifies the macroblock addresses mbAddrN and yM in two ordered steps:

1. Specification of a macroblock address mbAddrX depending on (xN, yN) and the following variables:
 - The variable currMbFrameFlag is derived as follows.
 - If the macroblock with address CurrMbAddr is a frame macroblock, currMbFrameFlag is set equal to 1,
 - Otherwise (the macroblock with address CurrMbAddr is a field macroblock), currMbFrameFlag is set equal to 0.
 - The variable mbIsTopMbFlag is derived as follows.
 - If the macroblock with address CurrMbAddr is a top macroblock (CurrMbAddr % 2 is equal to 0), mbIsTopMbFlag is set equal to 1;
 - Otherwise (the macroblock with address CurrMbAddr is a bottom macroblock, CurrMbAddr % 2 is equal to 1), mbIsTopMbFlag is set equal to 0.
2. Depending on the availability of mbAddrX, the following applies.
 - If mbAddrX is not available, mbAddrN is marked as not available.
 - Otherwise (mbAddrX is available), mbAddrN is marked as available and Table 6-4 specifies mbAddrN and yM depending on (xN, yN), currMbFrameFlag, mbIsTopMbFlag, and the variable mbAddrXFrameFlag, which is derived as follows.
 - If the macroblock mbAddrX is a frame macroblock, mbAddrXFrameFlag is set equal to 1,
 - Otherwise (the macroblock mbAddrX is a field macroblock), mbAddrXFrameFlag is set equal to 0.

Unspecified values (na) of the above flags in Table 6-4 indicate that the value of the corresponding flag is not relevant for the current table rows.

Table 6-4 – Specification of mbAddrN and yM

x_N	y_N	currMbFrameFlag	mbIsTopMbFlag	mbAddrX	mbAddrXFrameFlag	additional condition	mbAddrN	y_M
< 0	< 0	1	1	mbAddrD			mbAddrD + 1	y_N
			0	mbAddrA	1		mbAddrA	y_N
		0	1	mbAddrD	0		mbAddrA + 1	$(y_N + \text{maxH}) \gg 1$
			0	mbAddrD	1		mbAddrD + 1	$2 * y_N$
< 0	$0 \dots \text{maxH} - 1$	1	1	mbAddrA	1		mbAddrD	y_N
					0	$y_N \% 2 == 0$	mbAddrA	$y_N \gg 1$
					0	$y_N \% 2 \neq 0$	mbAddrA + 1	$y_N \gg 1$
			0	mbAddrA	1		mbAddrA + 1	y_N
					0	$y_N \% 2 == 0$	mbAddrA	$(y_N + \text{maxH}) \gg 1$
					0	$y_N \% 2 \neq 0$	mbAddrA + 1	$(y_N + \text{maxH}) \gg 1$
		0	1	mbAddrA	1	$y_N < (\text{maxH} / 2)$	mbAddrA	$y_N \ll 1$
					0	$y_N \geq (\text{maxH} / 2)$	mbAddrA + 1	$(y_N \ll 1) - \text{maxH}$
			0	mbAddrA	1		mbAddrA	y_N
					0	$y_N < (\text{maxH} / 2)$	mbAddrA	$(y_N \ll 1) + 1$
$0 \dots \text{maxW} - 1$	< 0	1	1	mbAddrB			mbAddrA + 1	y_N
			0	CurrMbAddr			CurrMbAddr - 1	y_N
		0	1	mbAddrB	1		mbAddrB + 1	$2 * y_N$
			0	mbAddrB	0		mbAddrB	y_N
$0 \dots \text{maxW} - 1$	$0 \dots \text{maxH} - 1$			CurrMbAddr			mbAddrB + 1	y_N
> $\text{maxW} - 1$	< 0	1	1	mbAddrC			CurrMbAddr	y_N
			0	not available			mbAddrC + 1	y_N
		0	1	mbAddrC	1		not available	na
			0	mbAddrC	0		mbAddrC + 1	$2 * y_N$
> $\text{maxW} - 1$	$0 \dots \text{maxH} - 1$			not available			mbAddrC	y_N
	> $\text{maxH} - 1$			not available			mbAddrC + 1	y_N
				not available			not available	na

The neighbouring luma location (x_W, y_W) relative to the upper-left corner of the macroblock mbAddrN is derived as

$$x_W = (x_N + \text{maxW}) \% \text{maxW} \quad (6-38)$$

$$y_W = (y_M + \text{maxH}) \% \text{maxH} \quad (6-39)$$

7 Syntax and semantics

7.1 Method of specifying syntax in tabular form

The syntax tables specify a superset of the syntax of all allowed bitstreams. Additional constraints on the syntax may be specified, either directly or indirectly, in other clauses.

NOTE – An actual decoder should implement means for identifying entry points into the bitstream and means to identify and handle non-conforming bitstreams. The methods for identifying and handling errors and other such situations are not specified here.

The following table lists examples of pseudo code used to describe the syntax. When **syntax_element** appears, it specifies that a syntax element is parsed from the bitstream and the bitstream pointer is advanced to the next position beyond the syntax element in the bitstream parsing process.

	C	Descriptor
/* A statement can be a syntax element with an associated syntax category and descriptor or can be an expression used to specify conditions for the existence, type, and quantity of syntax elements, as in the following two examples */		
syntax_element	3	ue(v)
conditioning statement		
/* A group of statements enclosed in curly brackets is a compound statement and is treated functionally as a single statement. */		
{		
statement		
statement		
...		
}		
/* A “while” structure specifies a test of whether a condition is true, and if true, specifies evaluation of a statement (or compound statement) repeatedly until the condition is no longer true */		
while(condition)		
statement		
/* A “do ... while” structure specifies evaluation of a statement once, followed by a test of whether a condition is true, and if true, specifies repeated evaluation of the statement until the condition is no longer true */		
do		
statement		
while(condition)		
/* An “if ... else” structure specifies a test of whether a condition is true, and if the condition is true, specifies evaluation of a primary statement, otherwise, specifies evaluation of an alternative statement. The “else” part of the structure and the associated alternative statement is omitted if no alternative statement evaluation is needed */		
if(condition)		
primary statement		
else		
alternative statement		
/* A “for” structure specifies evaluation of an initial statement, followed by a test of a condition, and if the condition is true, specifies repeated evaluation of a primary statement followed by a subsequent statement until the condition is no longer true. */		
for(initial statement; condition; subsequent statement)		
primary statement		

7.2 Specification of syntax functions, categories, and descriptors

The functions presented here are used in the syntactical description. These functions assume the existence of a bitstream pointer with an indication of the position of the next bit to be read by the decoding process from the bitstream.

`byte_aligned()` is specified as follows.

- If the current position in the bitstream is on a byte boundary, i.e., the next bit in the bitstream is the first bit in a byte, the return value of `byte_aligned()` is equal to TRUE.
- Otherwise, the return value of `byte_aligned()` is equal to FALSE.

`more_data_in_byte_stream()`, which is used only in the byte stream NAL unit syntax structure specified in Annex B, is specified as follows.

- If more data follow in the byte stream, the return value of `more_data_in_byte_stream()` is equal to TRUE.
- Otherwise, the return value of `more_data_in_byte_stream()` is equal to FALSE.

`more_rbsp_data()` is specified as follows.

- If there is more data in an RBSP before `rbsp_trailing_bits()`, the return value of `more_rbsp_data()` is equal to TRUE.
- Otherwise, the return value of `more_rbsp_data()` is equal to FALSE.

The method for enabling determination of whether there is more data in the RBSP is specified by the application (or in Annex B for applications that use the byte stream format).

`more_rbsp_trailing_data()` is specified as follows.

- If there is more data in an RBSP, the return value of `more_rbsp_trailing_data()` is equal to TRUE.
- Otherwise, the return value of `more_rbsp_trailing_data()` is equal to FALSE.

`next_bits(n)` provides the next bits in the bitstream for comparison purposes, without advancing the bitstream pointer. Provides a look at the next *n* bits in the bitstream with *n* being its argument. When used within the byte stream as specified in Annex B, `next_bits(n)` returns a value of 0 if fewer than *n* bits remain within the byte stream.

`read_bits(n)` reads the next *n* bits from the bitstream and advances the bitstream pointer by *n* bit positions. When *n* is equal to 0, `read_bits(n)` is specified to return a value equal to 0 and to not advance the bitstream pointer.

Categories (labelled in the table as C) specify the partitioning of slice data into at most three slice data partitions. Slice data partition A contains all syntax elements of category 2. Slice data partition B contains all syntax elements of category 3. Slice data partition C contains all syntax elements of category 4. The meaning of other category values is not specified. For some syntax elements, two category values, separated by a vertical bar, are used. In these cases, the category value to be applied is further specified in the text. For syntax structures used within other syntax structures, the categories of all syntax elements found within the included syntax structure are listed, separated by a vertical bar. A syntax element or syntax structure with category marked as "All" is present within all syntax structures that include that syntax element or syntax structure. For syntax structures used within other syntax structures, a numeric category value provided in a syntax table at the location of the inclusion of a syntax structure containing a syntax element with category marked as "All" is considered to apply to the syntax elements with category "All".

The following descriptors specify the parsing process of each syntax element. For some syntax elements, two descriptors, separated by a vertical bar, are used. In these cases, the left descriptors apply when `entropy_coding_mode_flag` is equal to 0 and the right descriptor applies when `entropy_coding_mode_flag` is equal to 1.

- `ae(v)`: context-adaptive arithmetic entropy-coded syntax element. The parsing process for this descriptor is specified in subclause 9.3.
- `b(8)`: byte having any pattern of bit string (8 bits). The parsing process for this descriptor is specified by the return value of the function `read_bits(8)`.
- `ce(v)`: context-adaptive variable-length entropy-coded syntax element with the left bit first. The parsing process for this descriptor is specified in subclause 9.2.
- `f(n)`: fixed-pattern bit string using *n* bits written (from left to right) with the left bit first. The parsing process for this descriptor is specified by the return value of the function `read_bits(n)`.
- `i(n)`: signed integer using *n* bits. When *n* is "v" in the syntax table, the number of bits varies in a manner dependent on the value of other syntax elements. The parsing process for this descriptor is specified by the

return value of the function `read_bits(n)` interpreted as a two's complement integer representation with most significant bit written first.

- `me(v)`: mapped Exp-Golomb-coded syntax element with the left bit first. The parsing process for this descriptor is specified in subclause 9.1.
- `se(v)`: signed integer Exp-Golomb-coded syntax element with the left bit first. The parsing process for this descriptor is specified in subclause 9.1.
- `te(v)`: truncated Exp-Golomb-coded syntax element with left bit first. The parsing process for this descriptor is specified in subclause 9.1.
- `u(n)`: unsigned integer using `n` bits. When `n` is "v" in the syntax table, the number of bits varies in a manner dependent on the value of other syntax elements. The parsing process for this descriptor is specified by the return value of the function `read_bits(n)` interpreted as a binary representation of an unsigned integer with most significant bit written first.
- `ue(v)`: unsigned integer Exp-Golomb-coded syntax element with the left bit first. The parsing process for this descriptor is specified in subclause 9.1.

7.3 Syntax in tabular form

7.3.1 NAL unit syntax

<code>nal_unit(NumBytesInNALunit) {</code>	C	Descriptor
forbidden_zero_bit	All	f(1)
nal_ref_idc	All	u(2)
nal_unit_type	All	u(5)
<code>NumBytesInRBSP = 0</code>		
<code>for(i = 1; i < NumBytesInNALunit; i++) {</code>		
<code>if(i + 2 < NumBytesInNALunit && next_bits(24) == 0x000003) {</code>		
rbsp_byte[NumBytesInRBSP++]	All	b(8)
rbsp_byte[NumBytesInRBSP++]	All	b(8)
<code>i += 2</code>		
emulation_prevention_three_byte /* equal to 0x03 */	All	f(8)
<code>} else</code>		
rbsp_byte[NumBytesInRBSP++]	All	b(8)
<code>}</code>		
<code>}</code>		

7.3.2 Raw byte sequence payloads and RBSP trailing bits syntax

7.3.2.1 Sequence parameter set RBSP syntax

seq_parameter_set_rbsp() {	C	Descriptor
profile_idc	0	u(8)
constraint_set0_flag	0	u(1)
constraint_set1_flag	0	u(1)
constraint_set2_flag	0	u(1)
constraint_set3_flag	0	u(1)
reserved_zero_4bits /* equal to 0 */	0	u(4)
level_idc	0	u(8)
seq_parameter_set_id	0	ue(v)
if(profile_idc == 100 profile_idc == 110 profile_idc == 122 profile_idc == 144) {		
chroma_format_idc	0	ue(v)
if(chroma_format_idc == 3)		
residual_colour_transform_flag	0	u(1)
bit_depth_luma_minus8	0	ue(v)
bit_depth_chroma_minus8	0	ue(v)
qpprime_y_zero_transform_bypass_flag	0	u(1)
seq_scaling_matrix_present_flag	0	u(1)
if(seq_scaling_matrix_present_flag)		
for(i = 0; i < 8; i++) {		
seq_scaling_list_present_flag[i]	0	u(1)
if(seq_scaling_list_present_flag[i])		
if(i < 6)		
scaling_list(ScalingList4x4[i], 16, UseDefaultScalingMatrix4x4Flag[i])	0	
else		
scaling_list(ScalingList8x8[i - 6], 64, UseDefaultScalingMatrix8x8Flag[i - 6])	0	
}		
}		
log2_max_frame_num_minus4	0	ue(v)
pic_order_cnt_type	0	ue(v)
if(pic_order_cnt_type == 0)		
log2_max_pic_order_cnt_lsb_minus4	0	ue(v)
else if(pic_order_cnt_type == 1) {		
delta_pic_order_always_zero_flag	0	u(1)
offset_for_non_ref_pic	0	se(v)
offset_for_top_to_bottom_field	0	se(v)
num_ref_frames_in_pic_order_cnt_cycle	0	ue(v)
for(i = 0; i < num_ref_frames_in_pic_order_cnt_cycle; i++)		
offset_for_ref_frame[i]	0	se(v)
}		
num_ref_frames	0	ue(v)
gaps_in_frame_num_value_allowed_flag	0	u(1)
pic_width_in_mbs_minus1	0	ue(v)
pic_height_in_map_units_minus1	0	ue(v)

frame_mbs_only_flag	0	u(1)
if(!frame_mbs_only_flag)		
mb_adaptive_frame_field_flag	0	u(1)
direct_8x8_inference_flag	0	u(1)
frame_cropping_flag	0	u(1)
if(frame_cropping_flag) {		
frame_crop_left_offset	0	ue(v)
frame_crop_right_offset	0	ue(v)
frame_crop_top_offset	0	ue(v)
frame_crop_bottom_offset	0	ue(v)
}		
vui_parameters_present_flag	0	u(1)
if(vui_parameters_present_flag)		
vui_parameters()	0	
rbsp_trailing_bits()	0	
}		

7.3.2.1.1 Scaling list syntax

scaling_list(scalingList, sizeOfScalingList, useDefaultScalingMatrixFlag) {	C	Descriptor
lastScale = 8		
nextScale = 8		
for(j = 0; j < sizeOfScalingList; j++) {		
if(nextScale != 0) {		
delta_scale	0 1	se(v)
nextScale = (lastScale + delta_scale + 256) % 256		
useDefaultScalingMatrixFlag = (j == 0 && nextScale == 0)		
}		
scalingList[j] = (nextScale == 0) ? lastScale : nextScale		
lastScale = scalingList[j]		
}		
}		

7.3.2.1.2 Sequence parameter set extension RBSP syntax

seq_parameter_set_extension_rbsp() {	C	Descriptor
seq_parameter_set_id	10	ue(v)
aux_format_idc	10	ue(v)
if(aux_format_idc != 0) {		
bit_depth_aux_minus8	10	ue(v)
alpha_incr_flag	10	u(1)
alpha_opaque_value	10	u(v)
alpha_transparent_value	10	u(v)
}		
additional_extension_flag	10	u(1)
rbsp_trailing_bits()	10	
}		

7.3.2.2 Picture parameter set RBSP syntax

pic_parameter_set_rbsp() {	C	Descriptor
pic_parameter_set_id	1	ue(v)
seq_parameter_set_id	1	ue(v)
entropy_coding_mode_flag	1	u(1)
pic_order_present_flag	1	u(1)
num_slice_groups_minus1	1	ue(v)
if(num_slice_groups_minus1 > 0) {		
slice_group_map_type	1	ue(v)
if(slice_group_map_type == 0)		
for(iGroup = 0; iGroup <= num_slice_groups_minus1; iGroup++)		
run_length_minus1 [iGroup]	1	ue(v)
else if(slice_group_map_type == 2)		
for(iGroup = 0; iGroup < num_slice_groups_minus1; iGroup++) {		
top_left [iGroup]	1	ue(v)
bottom_right [iGroup]	1	ue(v)
}		
else if(slice_group_map_type == 3		
slice_group_map_type == 4		
slice_group_map_type == 5) {		
slice_group_change_direction_flag	1	u(1)
slice_group_change_rate_minus1	1	ue(v)
} else if(slice_group_map_type == 6) {		
pic_size_in_map_units_minus1	1	ue(v)
for(i = 0; i <= pic_size_in_map_units_minus1; i++)		
slice_group_id [i]	1	u(v)
}		
}		
num_ref_idx_l0_active_minus1	1	ue(v)
num_ref_idx_l1_active_minus1	1	ue(v)
weighted_pred_flag	1	u(1)
weighted_bipred_idc	1	u(2)
pic_init_qp_minus26 /* relative to 26 */	1	se(v)
pic_init_qs_minus26 /* relative to 26 */	1	se(v)
chroma_qp_index_offset	1	se(v)
deblocking_filter_control_present_flag	1	u(1)
constrained_intra_pred_flag	1	u(1)
redundant_pic_cnt_present_flag	1	u(1)
if(more_rbsp_data()) {		
transform_8x8_mode_flag	1	u(1)
pic_scaling_matrix_present_flag	1	u(1)
if(pic_scaling_matrix_present_flag)		
for(i = 0; i < 6 + 2* transform_8x8_mode_flag; i++) {		
pic_scaling_list_present_flag [i]	1	u(1)
if(pic_scaling_list_present_flag[i])		
if(i < 6)		

scaling_list(ScalingList4x4[i], 16, UseDefaultScalingMatrix4x4Flag[i])	1	
else		
scaling_list(ScalingList8x8[i – 6], 64, UseDefaultScalingMatrix8x8Flag[i – 6])	1	
}		
second_chroma_qp_index_offset	1	se(v)
}		
rbsp_trailing_bits()	1	
}		

7.3.2.3 Supplemental enhancement information RBSP syntax

sei_rbsp() {	C	Descriptor
do		
sei_message()	5	
while(more_rbsp_data())		
rbsp_trailing_bits()	5	
}		

7.3.2.3.1 Supplemental enhancement information message syntax

sei_message() {	C	Descriptor
payloadType = 0		
while(next_bits(8) == 0xFF) {		
ff_byte /* equal to 0xFF */	5	f(8)
payloadType += 255		
}		
last_payload_type_byte	5	u(8)
payloadType += last_payload_type_byte		
payloadSize = 0		
while(next_bits(8) == 0xFF) {		
ff_byte /* equal to 0xFF */	5	f(8)
payloadSize += 255		
}		
last_payload_size_byte	5	u(8)
payloadSize += last_payload_size_byte		
sei_payload(payloadType, payloadSize)	5	
}		

7.3.2.4 Access unit delimiter RBSP syntax

access_unit_delimiter_rbsp() {	C	Descriptor
primary_pic_type	6	u(3)
rbsp_trailing_bits()	6	
}		

7.3.2.5 End of sequence RBSP syntax

end_of_seq_rbsp() {	C	Descriptor
}		

7.3.2.6 End of stream RBSP syntax

end_of_stream_rbsp() {	C	Descriptor
}		

7.3.2.7 Filler data RBSP syntax

filler_data_rbsp() {	C	Descriptor
while(next_bits(8) == 0xFF)		
ff_byte /* equal to 0xFF */	9	f(8)
rbsp_trailing_bits()	9	
}		

7.3.2.8 Slice layer without partitioning RBSP syntax

slice_layer_without_partitioning_rbsp() {	C	Descriptor
slice_header()	2	
slice_data() /* all categories of slice_data() syntax */	2 3 4	
rbsp_slice_trailing_bits()	2	
}		

7.3.2.9 Slice data partition RBSP syntax

7.3.2.9.1 Slice data partition A RBSP syntax

slice_data_partition_a_layer_rbsp() {	C	Descriptor
slice_header()	2	
slice_id	All	ue(v)
slice_data() /* only category 2 parts of slice_data() syntax */	2	
rbsp_slice_trailing_bits()	2	
}		

7.3.2.9.2 Slice data partition B RBSP syntax

slice_data_partition_b_layer_rbsp() {	C	Descriptor
slice_id	All	ue(v)
if(redundant_pic_cnt_present_flag)		
redundant_pic_cnt	All	ue(v)
slice_data() /* only category 3 parts of slice_data() syntax */	3	
rbsp_slice_trailing_bits()	3	
}		

7.3.2.9.3 Slice data partition C RBSP syntax

slice_data_partition_c_layer_rbsp() {	C	Descriptor
slice_id	All	ue(v)
if(redundant_pic_cnt_present_flag)		
redundant_pic_cnt	All	ue(v)
slice_data() /* only category 4 parts of slice_data() syntax */	4	
rbsp_slice_trailing_bits()	4	
}		

7.3.2.10 RBSP slice trailing bits syntax

rbsp_slice_trailing_bits() {	C	Descriptor
rbsp_trailing_bits()	All	
if(entropy_coding_mode_flag)		
while(more_rbsp_trailing_data())		
cabac_zero_word /* equal to 0x0000 */	All	f(16)
}		

7.3.2.11 RBSP trailing bits syntax

rbsp_trailing_bits() {	C	Descriptor
rbsp_stop_one_bit /* equal to 1 */	All	f(1)
while(!byte_aligned())		
rbsp_alignment_zero_bit /* equal to 0 */	All	f(1)
}		

7.3.3 Slice header syntax

slice_header() {	C	Descriptor
first_mb_in_slice	2	ue(v)
slice_type	2	ue(v)
pic_parameter_set_id	2	ue(v)
frame_num	2	u(v)
if(!frame_mbs_only_flag) {		
field_pic_flag	2	u(1)
if(field_pic_flag)		
bottom_field_flag	2	u(1)
}		
if(nal_unit_type == 5)		
idr_pic_id	2	ue(v)
if(pic_order_cnt_type == 0) {		
pic_order_cnt_lsb	2	u(v)
if(pic_order_present_flag && !field_pic_flag)		
delta_pic_order_cnt_bottom	2	se(v)
}		
if(pic_order_cnt_type == 1 && !delta_pic_order_always_zero_flag) {		
delta_pic_order_cnt[0]	2	se(v)
if(pic_order_present_flag && !field_pic_flag)		
delta_pic_order_cnt[1]	2	se(v)
}		
if(redundant_pic_cnt_present_flag)		
redundant_pic_cnt	2	ue(v)
if(slice_type == B)		
direct_spatial_mv_pred_flag	2	u(1)
if(slice_type == P slice_type == SP slice_type == B) {		
num_ref_idx_active_override_flag	2	u(1)
if(num_ref_idx_active_override_flag) {		
num_ref_idx_l0_active_minus1	2	ue(v)
if(slice_type == B)		
num_ref_idx_l1_active_minus1	2	ue(v)
}		
}		
ref_pic_list_reordering()	2	
if((weighted_pred_flag && (slice_type == P slice_type == SP)) (weighted_bipred_idc == 1 && slice_type == B))		
pred_weight_table()	2	
if(nal_ref_idc != 0)		
dec_ref_pic_marking()	2	
if(entropy_coding_mode_flag && slice_type != I && slice_type != SI)		
cabac_init_idc	2	ue(v)
slice_qp_delta	2	se(v)
if(slice_type == SP slice_type == SI) {		
if(slice_type == SP)		
sp_for_switch_flag	2	u(1)

slice_qs_delta	2	se(v)
}		
if(deblocking_filter_control_present_flag) {		
disable_deblocking_filter_idc	2	ue(v)
if(disable_deblocking_filter_idc != 1) {		
slice_alpha_c0_offset_div2	2	se(v)
slice_beta_offset_div2	2	se(v)
}		
}		
if(num_slice_groups_minus1 > 0 && slice_group_map_type >= 3 && slice_group_map_type <= 5)		
slice_group_change_cycle	2	u(v)
}		

7.3.3.1 Reference picture list reordering syntax

ref_pic_list_reordering() {	C	Descriptor
if(slice_type != I && slice_type != SI) {		
ref_pic_list_reordering_flag_l0	2	u(1)
if(ref_pic_list_reordering_flag_l0)		
do {		
reordering_of_pic_nums_idc	2	ue(v)
if(reordering_of_pic_nums_idc == 0 reordering_of_pic_nums_idc == 1)		
abs_diff_pic_num_minus1	2	ue(v)
else if(reordering_of_pic_nums_idc == 2)		
long_term_pic_num	2	ue(v)
} while(reordering_of_pic_nums_idc != 3)		
}		
if(slice_type == B) {		
ref_pic_list_reordering_flag_l1	2	u(1)
if(ref_pic_list_reordering_flag_l1)		
do {		
reordering_of_pic_nums_idc	2	ue(v)
if(reordering_of_pic_nums_idc == 0 reordering_of_pic_nums_idc == 1)		
abs_diff_pic_num_minus1	2	ue(v)
else if(reordering_of_pic_nums_idc == 2)		
long_term_pic_num	2	ue(v)
} while(reordering_of_pic_nums_idc != 3)		
}		
}		

7.3.3.2 Prediction weight table syntax

pred_weight_table() {	C	Descriptor
luma_log2_weight_denom	2	ue(v)
if(chroma_format_idc != 0)		
chroma_log2_weight_denom	2	ue(v)
for(i = 0; i <= num_ref_idx_l0_active_minus1; i++) {		
luma_weight_l0_flag	2	u(1)
if(luma_weight_l0_flag) {		
luma_weight_l0[i]	2	se(v)
luma_offset_l0[i]	2	se(v)
}		
if (chroma_format_idc != 0) {		
chroma_weight_l0_flag	2	u(1)
if(chroma_weight_l0_flag)		
for(j = 0; j < 2; j++) {		
chroma_weight_l0[i][j]	2	se(v)
chroma_offset_l0[i][j]	2	se(v)
}		
}		
}		
if(slice_type == B)		
for(i = 0; i <= num_ref_idx_l1_active_minus1; i++) {		
luma_weight_l1_flag	2	u(1)
if(luma_weight_l1_flag) {		
luma_weight_l1[i]	2	se(v)
luma_offset_l1[i]	2	se(v)
}		
if(chroma_format_idc != 0) {		
chroma_weight_l1_flag	2	u(1)
if(chroma_weight_l1_flag)		
for(j = 0; j < 2; j++) {		
chroma_weight_l1[i][j]	2	se(v)
chroma_offset_l1[i][j]	2	se(v)
}		
}		
}		
}		

7.3.3.3 Decoded reference picture marking syntax

dec_ref_pic_marking() {	C	Descriptor
if(nal_unit_type == 5) {		
no_output_of_prior_pics_flag	2 5	u(1)
long_term_reference_flag	2 5	u(1)
} else {		
adaptive_ref_pic_marking_mode_flag	2 5	u(1)
if(adaptive_ref_pic_marking_mode_flag)		
do {		
memory_management_control_operation	2 5	ue(v)
if(memory_management_control_operation == 1 memory_management_control_operation == 3)		
difference_of_pic_nums_minus1	2 5	ue(v)
if(memory_management_control_operation == 2)		
long_term_pic_num	2 5	ue(v)
if(memory_management_control_operation == 3 memory_management_control_operation == 6)		
long_term_frame_idx	2 5	ue(v)
if(memory_management_control_operation == 4)		
max_long_term_frame_idx_plus1	2 5	ue(v)
} while(memory_management_control_operation != 0)		
}		
}		

7.3.4 Slice data syntax

slice_data() {	C	Descriptor
if(entropy_coding_mode_flag)		
while(!byte_aligned())		
cabac_alignment_one_bit	2	f(1)
CurrMbAddr = first_mb_in_slice * (1 + MbaffFrameFlag)		
moreDataFlag = 1		
prevMbSkipped = 0		
do {		
if(slice_type != I && slice_type != SI)		
if(!entropy_coding_mode_flag) {		
mb_skip_run	2	ue(v)
prevMbSkipped = (mb_skip_run > 0)		
for(i=0; i<mb_skip_run; i++)		
CurrMbAddr = NextMbAddress(CurrMbAddr)		
moreDataFlag = more_rbsp_data()		
} else {		
mb_skip_flag	2	ae(v)
moreDataFlag = !mb_skip_flag		
}		
if(moreDataFlag) {		
if(MbaffFrameFlag && (CurrMbAddr % 2 == 0 (CurrMbAddr % 2 == 1 && prevMbSkipped)))		
mb_field_decoding_flag	2	u(1) ae(v)
macroblock_layer()	2 3 4	
}		
if(!entropy_coding_mode_flag)		
moreDataFlag = more_rbsp_data()		
else {		
if(slice_type != I && slice_type != SI)		
prevMbSkipped = mb_skip_flag		
if(MbaffFrameFlag && CurrMbAddr % 2 == 0)		
moreDataFlag = 1		
else {		
end_of_slice_flag	2	ae(v)
moreDataFlag = !end_of_slice_flag		
}		
}		
CurrMbAddr = NextMbAddress(CurrMbAddr)		
} while(moreDataFlag)		
}		

7.3.5 Macroblock layer syntax

macroblock_layer() {	C	Descriptor
mb_type	2	ue(v) ae(v)
if(mb_type == I_PCM) {		
while(!byte_aligned())		
pcm_alignment_zero_bit	2	f(1)
for(i = 0; i < 256; i++)		
pcm_sample_luma[i]	2	u(v)
for(i = 0; i < 2 * MbWidthC * MbHeightC; i++)		
pcm_sample_chroma[i]	2	u(v)
} else {		
noSubMbPartSizeLessThan8x8Flag = 1		
if(mb_type != I_NxN && MbPartPredMode(mb_type, 0) != Intra_16x16 && NumMbPart(mb_type) == 4) {		
sub_mb_pred(mb_type)	2	
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(sub_mb_type[mbPartIdx] != B_Direct_8x8) {		
if(NumSubMbPart(sub_mb_type[mbPartIdx]) > 1)		
noSubMbPartSizeLessThan8x8Flag = 0		
} else if(!direct_8x8_inference_flag)		
noSubMbPartSizeLessThan8x8Flag = 0		
} else {		
if(transform_8x8_mode_flag && mb_type == I_NxN)		
transform_size_8x8_flag	2	u(1) ae(v)
mb_pred(mb_type)	2	
}		
if(MbPartPredMode(mb_type, 0) != Intra_16x16) {		
coded_block_pattern	2	me(v) ae(v)
if(CodedBlockPatternLuma > 0 && transform_8x8_mode_flag && mb_type != I_NxN && noSubMbPartSizeLessThan8x8Flag && (mb_type != B_Direct_16x16 direct_8x8_inference_flag))		
transform_size_8x8_flag	2	u(1) ae(v)
}		
if(CodedBlockPatternLuma > 0 CodedBlockPatternChroma > 0 MbPartPredMode(mb_type, 0) == Intra_16x16) {		
mb_qp_delta	2	se(v) ae(v)
residual()	3 4	
}		
}		
}		

7.3.5.1 Macroblock prediction syntax

mb_pred(mb_type) {	C	Descriptor
if(MbPartPredMode(mb_type, 0) == Intra_4x4 MbPartPredMode(mb_type, 0) == Intra_8x8 MbPartPredMode(mb_type, 0) == Intra_16x16) {		
if(MbPartPredMode(mb_type, 0) == Intra_4x4)		
for(luma4x4BlkIdx=0; luma4x4BlkIdx<16; luma4x4BlkIdx++) {		
prev_intra4x4_pred_mode_flag [luma4x4BlkIdx]	2	u(1) ae(v)
if(!prev_intra4x4_pred_mode_flag[luma4x4BlkIdx])		
rem_intra4x4_pred_mode [luma4x4BlkIdx]	2	u(3) ae(v)
}		
if(MbPartPredMode(mb_type, 0) == Intra_8x8)		
for(luma8x8BlkIdx=0; luma8x8BlkIdx<4; luma8x8BlkIdx++) {		
prev_intra8x8_pred_mode_flag [luma8x8BlkIdx]	2	u(1) ae(v)
if(!prev_intra8x8_pred_mode_flag[luma8x8BlkIdx])		
rem_intra8x8_pred_mode [luma8x8BlkIdx]	2	u(3) ae(v)
}		
if(chroma_format_idc != 0)		
intra_chroma_pred_mode	2	ue(v) ae(v)
} else if(MbPartPredMode(mb_type, 0) != Direct) {		
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if((num_ref_idx_l0_active_minus1 > 0 mb_field_decoding_flag) && MbPartPredMode(mb_type, mbPartIdx) != Pred_L1)		
ref_idx_l0 [mbPartIdx]	2	te(v) ae(v)
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if((num_ref_idx_l1_active_minus1 > 0 mb_field_decoding_flag) && MbPartPredMode(mb_type, mbPartIdx) != Pred_L0)		
ref_idx_l1 [mbPartIdx]	2	te(v) ae(v)
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if(MbPartPredMode(mb_type, mbPartIdx) != Pred_L1)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0 [mbPartIdx][0][compIdx]	2	se(v) ae(v)
for(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
if(MbPartPredMode(mb_type, mbPartIdx) != Pred_L0)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1 [mbPartIdx][0][compIdx]	2	se(v) ae(v)
}		
}		

7.3.5.2 Sub-macroblock prediction syntax

sub_mb_pred(mb_type) {	C	Descriptor
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
sub_mb_type [mbPartIdx]	2	ue(v) ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if((num_ref_idx_l0_active_minus1 > 0 mb_field_decoding_flag) && mb_type != P_8x8ref0 && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L1)		
ref_idx_l0 [mbPartIdx]	2	te(v) ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if((num_ref_idx_l1_active_minus1 > 0 mb_field_decoding_flag) && sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L0)		
ref_idx_l1 [mbPartIdx]	2	te(v) ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L1)		
for(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub_mb_type[mbPartIdx]); subMbPartIdx++)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0 [mbPartIdx][subMbPartIdx][compIdx]	2	se(v) ae(v)
for(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
if(sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L0)		
for(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub_mb_type[mbPartIdx]); subMbPartIdx++)		
for(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1 [mbPartIdx][subMbPartIdx][compIdx]	2	se(v) ae(v)
}		

7.3.5.3 Residual data syntax

residual() {	C	Descriptor
if(!entropy_coding_mode_flag)		
residual_block = residual_block_cavlc		
else		
residual_block = residual_block_cabac		
if(MbPartPredMode(mb_type, 0) == Intra_16x16)		
residual_block(Intra16x16DCLevel, 16)	3	
for(i8x8 = 0; i8x8 < 4; i8x8++) /* each luma 8x8 block */		
if(!transform_size_8x8_flag !entropy_coding_mode_flag)		
for(i4x4 = 0; i4x4 < 4; i4x4++) { /* each 4x4 sub-block of block */		
if(CodedBlockPatternLuma & (1 << i8x8))		
if(MbPartPredMode(mb_type, 0) == Intra_16x16)		
residual_block(Intra16x16ACLevel[i8x8 * 4 + i4x4], 15)	3	
else		
residual_block(LumaLevel[i8x8 * 4 + i4x4], 16)	3 4	
else if(MbPartPredMode(mb_type, 0) == Intra_16x16)		
for(i = 0; i < 15; i++)		
Intra16x16ACLevel[i8x8 * 4 + i4x4][i] = 0		
else		
for(i = 0; i < 16; i++)		
LumaLevel[i8x8 * 4 + i4x4][i] = 0		
if(!entropy_coding_mode_flag && transform_size_8x8_flag)		
for(i = 0; i < 16; i++)		
LumaLevel8x8[i8x8][4 * i + i4x4] =		
LumaLevel[i8x8 * 4 + i4x4][i]		
}		
else if(CodedBlockPatternLuma & (1 << i8x8))		
residual_block(LumaLevel8x8[i8x8], 64)	3 4	
else		
for(i = 0; i < 64; i++)		
LumaLevel8x8[i8x8][i] = 0		
if(chroma_format_idc != 0) {		
NumC8x8 = 4 / (SubWidthC * SubHeightC)		
for(iCbCr = 0; iCbCr < 2; iCbCr++)		
if(CodedBlockPatternChroma & 3) /* chroma DC residual present */		
residual_block(ChromaDCLevel[iCbCr], 4 * NumC8x8)	3 4	
else		
for(i = 0; i < 4 * NumC8x8; i++)		
ChromaDCLevel[iCbCr][i] = 0		
for(iCbCr = 0; iCbCr < 2; iCbCr++)		
for(i8x8 = 0; i8x8 < NumC8x8; i8x8++)		
for(i4x4 = 0; i4x4 < 4; i4x4++)		
if(CodedBlockPatternChroma & 2)		
/* chroma AC residual present */		
residual_block(ChromaACLevel[iCbCr][i8x8*4+i4x4], 15)	3 4	
else		
for(i = 0; i < 15; i++)		
ChromaACLevel[iCbCr][i8x8*4+i4x4][i] = 0		
}		

7.3.5.3.1 Residual block CAVLC syntax

residual_block_cavlc(coeffLevel, maxNumCoeff) {	C	Descriptor
for(i = 0; i < maxNumCoeff; i++)		
coeffLevel[i] = 0		
coeff_token	3 4	ce(v)
if(TotalCoeff(coeff_token) > 0) {		
if(TotalCoeff(coeff_token) > 10 && TrailingOnes(coeff_token) < 3)		
suffixLength = 1		
else		
suffixLength = 0		
for(i = 0; i < TotalCoeff(coeff_token); i++)		
if(i < TrailingOnes(coeff_token)) {		
trailing_ones_sign_flag	3 4	u(1)
level[i] = 1 - 2 * trailing_ones_sign_flag		
} else {		
level_prefix	3 4	ce(v)
levelCode = (Min(15, level_prefix) << suffixLength)		
if(suffixLength > 0 level_prefix >= 14) {		
level_suffix	3 4	u(v)
levelCode += level_suffix		
}		
if(level_prefix >= 15 && suffixLength == 0)		
levelCode += 15		
if(level_prefix >= 16)		
levelCode += (1 << (level_prefix - 3)) - 4096		
if(i == TrailingOnes(coeff_token) && TrailingOnes(coeff_token) < 3)		
levelCode += 2		
if(levelCode % 2 == 0)		
level[i] = (levelCode + 2) >> 1		
else		
level[i] = (-levelCode - 1) >> 1		
if(suffixLength == 0)		
suffixLength = 1		
if(Abs(level[i]) > (3 << (suffixLength - 1)) && suffixLength < 6)		
suffixLength++		
}		
if(TotalCoeff(coeff_token) < maxNumCoeff) {		
total_zeros	3 4	ce(v)
zerosLeft = total_zeros		
} else		
zerosLeft = 0		
for(i = 0; i < TotalCoeff(coeff_token) - 1; i++) {		
if(zerosLeft > 0) {		
run_before	3 4	ce(v)
run[i] = run_before		
} else		
run[i] = 0		

zerosLeft = zerosLeft – run[i]		
}		
run[TotalCoeff(coeff_token) – 1] = zerosLeft		
coeffNum = -1		
for(i = TotalCoeff(coeff_token) – 1; i >= 0; i--) {		
coeffNum += run[i] + 1		
coeffLevel[coeffNum] = level[i]		
}		
}		
}		

7.3.5.3.2 Residual block CABAC syntax

residual_block_cabac(coeffLevel, maxNumCoeff) {	C	Descriptor
if(maxNumCoeff == 64)		
coded_block_flag = 1		
else		
coded_block_flag	3 4	ae(v)
if(coded_block_flag) {		
numCoeff = maxNumCoeff		
i = 0		
do {		
significant_coeff_flag[i]	3 4	ae(v)
if(significant_coeff_flag[i]) {		
last_significant_coeff_flag[i]	3 4	ae(v)
if(last_significant_coeff_flag[i]) {		
numCoeff = i + 1		
for(j = numCoeff; j < maxNumCoeff; j++)		
coeffLevel[j] = 0		
}		
}		
i++		
} while(i < numCoeff - 1)		
coeff_abs_level_minus1[numCoeff - 1]	3 4	ae(v)
coeff_sign_flag[numCoeff - 1]	3 4	ae(v)
coeffLevel[numCoeff - 1] =		
(coeff_abs_level_minus1[numCoeff - 1] + 1) *		
(1 – 2 * coeff_sign_flag[numCoeff - 1])		
for(i = numCoeff - 2; i >= 0; i--)		
if(significant_coeff_flag[i]) {		
coeff_abs_level_minus1[i]	3 4	ae(v)
coeff_sign_flag[i]	3 4	ae(v)
coeffLevel[i] = (coeff_abs_level_minus1[i] + 1) *		
(1 – 2 * coeff_sign_flag[i])		
} else		
coeffLevel[i] = 0		
} else		
for(i = 0; i < maxNumCoeff; i++)		
coeffLevel[i] = 0		
}		

7.4 Semantics

Semantics associated with the syntax structures and with the syntax elements within these structures are specified in this subclause. When the semantics of a syntax element are specified using a table or a set of tables, any values that are not specified in the table(s) shall not be present in the bitstream unless otherwise specified in this Recommendation | International Standard.

7.4.1 NAL unit semantics

NOTE 1 – The VCL is specified to efficiently represent the content of the video data. The NAL is specified to format that data and provide header information in a manner appropriate for conveyance on a variety of communication channels or storage media. All data are contained in NAL units, each of which contains an integer number of bytes. A NAL unit specifies a generic format for use in both packet-oriented and bitstream systems. The format of NAL units for both packet-oriented transport and byte stream is identical except that each NAL unit can be preceded by a start code prefix and extra padding bytes in the byte stream format.

NumBytesInNALunit specifies the size of the NAL unit in bytes. This value is required for decoding of the NAL unit. Some form of demarcation of NAL unit boundaries is necessary to enable inference of NumBytesInNALunit. One such demarcation method is specified in Annex B for the byte stream format. Other methods of demarcation may be specified outside of this Recommendation | International Standard.

forbidden_zero_bit shall be equal to 0.

nal_ref_idc not equal to 0 specifies that the content of the NAL unit contains a sequence parameter set or a picture parameter set or a slice of a reference picture or a slice data partition of a reference picture.

nal_ref_idc equal to 0 for a NAL unit containing a slice or slice data partition indicates that the slice or slice data partition is part of a non-reference picture.

nal_ref_idc shall not be equal to 0 for sequence parameter set or sequence parameter set extension or picture parameter set NAL units. When nal_ref_idc is equal to 0 for one slice or slice data partition NAL unit of a particular picture, it shall be equal to 0 for all slice and slice data partition NAL units of the picture.

nal_ref_idc shall not be equal to 0 for IDR NAL units, i.e., NAL units with nal_unit_type equal to 5.

nal_ref_idc shall be equal to 0 for all NAL units having nal_unit_type equal to 6, 9, 10, 11, or 12.

nal_unit_type specifies the type of RBSP data structure contained in the NAL unit as specified in Table 7-1. VCL NAL units are specified as those NAL units having nal_unit_type equal to 1 to 5, inclusive. All remaining NAL units are called non-VCL NAL units.

The column marked "C" in Table 7-1 lists the categories of the syntax elements that may be present in the NAL unit. In addition, syntax elements with syntax category "All" may be present, as determined by the syntax and semantics of the RBSP data structure. The presence or absence of any syntax elements of a particular listed category is determined from the syntax and semantics of the associated RBSP data structure. nal_unit_type shall not be equal to 3 or 4 unless at least one syntax element is present in the RBSP data structure having a syntax element category value equal to the value of nal_unit_type and not categorized as "All".

Table 7-1 – NAL unit type codes

nal_unit_type	Content of NAL unit and RBSP syntax structure	C
0	Unspecified	
1	Coded slice of a non-IDR picture slice_layer_without_partitioning_rbsp()	2, 3, 4
2	Coded slice data partition A slice_data_partition_a_layer_rbsp()	2
3	Coded slice data partition B slice_data_partition_b_layer_rbsp()	3
4	Coded slice data partition C slice_data_partition_c_layer_rbsp()	4
5	Coded slice of an IDR picture slice_layer_without_partitioning_rbsp()	2, 3
6	Supplemental enhancement information (SEI) sei_rbsp()	5
7	Sequence parameter set seq_parameter_set_rbsp()	0
8	Picture parameter set pic_parameter_set_rbsp()	1
9	Access unit delimiter access_unit_delimiter_rbsp()	6
10	End of sequence end_of_seq_rbsp()	7
11	End of stream end_of_stream_rbsp()	8
12	Filler data filler_data_rbsp()	9
13	Sequence parameter set extension seq_parameter_set_extension_rbsp()	10
14..18	Reserved	
19	Coded slice of an auxiliary coded picture without partitioning slice_layer_without_partitioning_rbsp()	2, 3, 4
20..23	Reserved	
24..31	Unspecified	

NAL units having nal_unit_type equal to 13 and 19 may be discarded by decoders without affecting the decoding process for NAL units having nal_unit_type not equal to 13 or 19 and without affecting conformance to this Recommendation | International Standard.

NAL units that use nal_unit_type equal to 0 or in the range of 24..31, inclusive, shall not affect the decoding process specified in this Recommendation | International Standard.

NOTE 2 – NAL unit types 0 and 24..31 may be used as determined by the application. No decoding process for these values of nal_unit_type is specified in this Recommendation | International Standard.

Decoders shall ignore (remove from the bitstream and discard) the contents of all NAL units that use reserved values of nal_unit_type.

NOTE 3 – This requirement allows future definition of compatible extensions to this Recommendation | International Standard.

In the text, coded slice NAL unit collectively refers to a coded slice of a non-IDR picture NAL unit or to a coded slice of an IDR picture NAL unit.

When the value of `nal_unit_type` is equal to 5 for a NAL unit containing a slice of a coded picture, the value of `nal_unit_type` shall be 5 in all other VCL NAL units of the same coded picture. Such a picture is referred to as an IDR picture.

NOTE 4 – Slice data partitioning cannot be used for IDR pictures.

`rbsp_byte[i]` is the *i*-th byte of an RBSP. An RBSP is specified as an ordered sequence of bytes as follows.

The RBSP contains an SODB as follows.

- If the SODB is empty (i.e., zero bits in length), the RBSP is also empty.
- Otherwise, the RBSP contains the SODB as follows.
 - 1) The first byte of the RBSP contains the (most significant, left-most) eight bits of the SODB; the next byte of the RBSP shall contain the next eight bits of the SODB, etc., until fewer than eight bits of the SODB remain.
 - 2) `rbsp_trailing_bits()` are present after the SODB as follows:
 - i) The first (most significant, left-most) bits of the final RBSP byte contains the remaining bits of the SODB, (if any)
 - ii) The next bit consists of a single `rbsp_stop_one_bit` equal to 1, and
 - iii) When the `rbsp_stop_one_bit` is not the last bit of a byte-aligned byte, one or more `rbsp_alignment_zero_bit` is present to result in byte alignment.
 - 3) One or more `cabac_zero_word` 16-bit syntax elements equal to 0x0000 may be present in some RBSPs after the `rbsp_trailing_bits()` at the end of the RBSP.

Syntax structures having these RBSP properties are denoted in the syntax tables using an "`_rbsp`" suffix. These structures shall be carried within NAL units as the content of the `rbsp_byte[i]` data bytes. The association of the RBSP syntax structures to the NAL units shall be as specified in Table 7-1.

NOTE 5 – When the boundaries of the RBSP are known, the decoder can extract the SODB from the RBSP by concatenating the bits of the bytes of the RBSP and discarding the `rbsp_stop_one_bit`, which is the last (least significant, right-most) bit equal to 1, and discarding any following (less significant, farther to the right) bits that follow it, which are equal to 0. The data necessary for the decoding process is contained in the SODB part of the RBSP.

`emulation_prevention_three_byte` is a byte equal to 0x03. When an `emulation_prevention_three_byte` is present in the NAL unit, it shall be discarded by the decoding process.

The last byte of the NAL unit shall not be equal to 0x00.

Within the NAL unit, the following three-byte sequences shall not occur at any byte-aligned position:

- 0x000000
- 0x000001
- 0x000002

Within the NAL unit, any four-byte sequence that starts with 0x000003 other than the following sequences shall not occur at any byte-aligned position:

- 0x00000300
- 0x00000301
- 0x00000302
- 0x00000303

NOTE 6 – When `nal_unit_type` is equal to 0, particular care must be exercised in the design of encoders to avoid the presence of the above-listed three-byte and four-byte patterns at the beginning of the NAL unit syntax structure, as the syntax element `emulation_prevention_three_byte` cannot be the third byte of a NAL unit.

7.4.1.1 Encapsulation of an SODB within an RBSP (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

The form of encapsulation of an SODB within an RBSP and the use of the `emulation_prevention_three_byte` for encapsulation of an RBSP within a NAL unit is specified for the following purposes:

- to prevent the emulation of start codes within NAL units while allowing any arbitrary SODB to be represented within a NAL unit,
- to enable identification of the end of the SODB within the NAL unit by searching the RBSP for the `rbsp_stop_one_bit` starting at the end of the RBSP, and

- to enable a NAL unit to have a size larger than that of the SODB under some circumstances (using one or more `cabac_zero_word`).

The encoder can produce a NAL unit from an RBSP by the following procedure:

The RBSP data is searched for byte-aligned bits of the following binary patterns:

'00000000 00000000 000000xx' (where xx represents any 2 bit pattern: 00, 01, 10, or 11),

and a byte equal to 0x03 is inserted to replace these bit patterns with the patterns

'00000000 00000000 00000011 000000xx',

and finally, when the last byte of the RBSP data is equal to 0x00 (which can only occur when the RBSP ends in a `cabac_zero_word`), a final byte equal to 0x03 is appended to the end of the data.

The resulting sequence of bytes is then prefixed with the first byte of the NAL unit containing the indication of the type of RBSP data structure it contains. This results in the construction of the entire NAL unit.

This process can allow any SODB to be represented in a NAL unit while ensuring that

- no byte-aligned start code prefix is emulated within the NAL unit, and
- no sequence of 8 zero-valued bits followed by a start code prefix, regardless of byte-alignment, is emulated within the NAL unit.

7.4.1.2 Order of NAL units and association to coded pictures, access units, and video sequences

This subclause specifies constraints on the order of NAL units in the bitstream. Any order of NAL units in the bitstream obeying these constraints is referred to in the text as the decoding order of NAL units. Within a NAL unit, the syntax in subclauses 7.3, D.1, and E.1 specifies the decoding order of syntax elements. Decoders conforming to this Recommendation | International Standard shall be capable of receiving NAL units and their syntax elements in decoding order.

7.4.1.2.1 Order of sequence and picture parameter set RBSPs and their activation

NOTE 1 – The sequence and picture parameter set mechanism decouples the transmission of infrequently changing information from the transmission of coded macroblock data. Sequence and picture parameter sets may, in some applications, be conveyed "out-of-band" using a reliable transport mechanism.

A picture parameter set RBSP includes parameters that can be referred to by the coded slice NAL units or coded slice data partition A NAL units of one or more coded pictures. Each picture parameter set RBSP is initially considered not active at the start of the operation of the decoding process. At most one picture parameter set RBSP is considered active at any given moment during the operation of the decoding process, and the activation of any particular picture parameter set RBSP results in the deactivation of the previously-active picture parameter set RBSP (if any).

When a picture parameter set RBSP (with a particular value of `pic_parameter_set_id`) is not active and it is referred to by a coded slice NAL unit or coded slice data partition A NAL unit (using that value of `pic_parameter_set_id`), it is activated. This picture parameter set RBSP is called the active picture parameter set RBSP until it is deactivated by the activation of another picture parameter set RBSP. A picture parameter set RBSP, with that particular value of `pic_parameter_set_id`, shall be available to the decoding process prior to its activation.

Any picture parameter set NAL unit containing the value of `pic_parameter_set_id` for the active picture parameter set RBSP shall have the same content as that of the active picture parameter set RBSP unless it follows the last VCL NAL unit of a coded picture and precedes the first VCL NAL unit of another coded picture.

A sequence parameter set RBSP includes parameters that can be referred to by one or more picture parameter set RBSPs or one or more SEI NAL units containing a buffering period SEI message. Each sequence parameter set RBSP is initially considered not active at the start of the operation of the decoding process. At most one sequence parameter set RBSP is considered active at any given moment during the operation of the decoding process, and the activation of any particular sequence parameter set RBSP results in the deactivation of the previously-active sequence parameter set RBSP (if any).

When a sequence parameter set RBSP (with a particular value of `seq_parameter_set_id`) is not already active and it is referred to by activation of a picture parameter set RBSP (using that value of `seq_parameter_set_id`) or is referred to by an SEI NAL unit containing a buffering period SEI message (using that value of `seq_parameter_set_id`), it is activated. This sequence parameter set RBSP is called the active sequence parameter set RBSP until it is deactivated by the activation of another sequence parameter set RBSP. A sequence parameter set RBSP, with that particular value of `seq_parameter_set_id`, shall be available to the decoding process prior to its activation. An activated sequence parameter set RBSP shall remain active for the entire coded video sequence.

NOTE 2 – Because an IDR access unit begins a new coded video sequence and an activated sequence parameter set RBSP must remain active for the entire coded video sequence, a sequence parameter set RBSP can only be activated by a buffering period SEI message when the buffering period SEI message is part of an IDR access unit.

Any sequence parameter set NAL unit containing the value of `seq_parameter_set_id` for the active sequence parameter set RBSP shall have the same content as that of the active sequence parameter set RBSP unless it follows the last access unit of a coded video sequence and precedes the first VCL NAL unit and the first SEI NAL unit containing a buffering period SEI message (when present) of another coded video sequence.

NOTE 3 – If picture parameter set RBSP or sequence parameter set RBSP are conveyed within the bitstream, these constraints impose an order constraint on the NAL units that contain the picture parameter set RBSP or sequence parameter set RBSP, respectively. Otherwise (picture parameter set RBSP or sequence parameter set RBSP are conveyed by other means not specified in this Recommendation | International Standard), they must be available to the decoding process in a timely fashion such that these constraints are obeyed.

When present, a sequence parameter set extension RBSP includes parameters having a similar function to those of a sequence parameter set RBSP. For purposes of establishing constraints on the syntax elements of the sequence parameter set extension RBSP and for purposes of determining activation of a sequence parameter set extension RBSP, the sequence parameter set extension RBSP shall be considered part of the preceding sequence parameter set RBSP with the same value of `seq_parameter_set_id`. When a sequence parameter set RBSP is present that is not followed by a sequence parameter set extension RBSP with the same value of `seq_parameter_set_id` prior to the activation of the sequence parameter set RBSP, the sequence parameter set extension RBSP and its syntax elements shall be considered not present for the active sequence parameter set RBSP.

All constraints that are expressed on the relationship between the values of the syntax elements (and the values of variables derived from those syntax elements) in sequence parameter sets and picture parameter sets and other syntax elements are expressions of constraints that apply only to the active sequence parameter set and the active picture parameter set. If any sequence parameter set RBSP is present that is not activated in the bitstream, its syntax elements shall have values that would conform to the specified constraints if it were activated by reference in an otherwise-conforming bitstream. If any picture parameter set RBSP is present that is not ever activated in the bitstream, its syntax elements shall have values that would conform to the specified constraints if it were activated by reference in an otherwise-conforming bitstream.

During operation of the decoding process (see clause 8), the values of parameters of the active picture parameter set and the active sequence parameter set shall be considered in effect. For interpretation of SEI messages, the values of the parameters of the picture parameter set and sequence parameter set that are active for the operation of the decoding process for the VCL NAL units of the primary coded picture in the same access unit shall be considered in effect unless otherwise specified in the SEI message semantics.

7.4.1.2.2 Order of access units and association to coded video sequences

A bitstream conforming to this Recommendation | International Standard consists of one or more coded video sequences.

A coded video sequence consists of one or more access units. The order of NAL units and coded pictures and their association to access units is described in subclause 7.4.1.2.3.

The first access unit of each coded video sequence is an IDR access unit. All subsequent access units in the coded video sequence are non-IDR access units.

The values of picture order count for the coded pictures in consecutive access units in decoding order containing non-reference pictures shall be non-decreasing.

When present, an access unit following an access unit that contains an end of sequence NAL unit shall be an IDR access unit.

When an SEI NAL unit contains data that pertain to more than one access unit (for example, when the SEI NAL unit has a coded video sequence as its scope), it shall be contained in the first access unit to which it applies.

When an end of stream NAL unit is present in an access unit, this access unit shall be the last access unit in the bitstream and the end of stream NAL unit shall be the last NAL unit in that access unit.

7.4.1.2.3 Order of NAL units and coded pictures and association to access units

An access unit consists of one primary coded picture, zero or more corresponding redundant coded pictures, and zero or more non-VCL NAL units. The association of VCL NAL units to primary or redundant coded pictures is described in subclause 7.4.1.2.5.

The first access unit in the bitstream starts with the first NAL unit of the bitstream.

The first of any of the following NAL units after the last VCL NAL unit of a primary coded picture specifies the start of a new access unit.

- access unit delimiter NAL unit (when present)
- sequence parameter set NAL unit (when present)
- picture parameter set NAL unit (when present)
- SEI NAL unit (when present)
- NAL units with `nal_unit_type` in the range of 14 to 18, inclusive
- first VCL NAL unit of a primary coded picture (always present)

The constraints for the detection of the first VCL NAL unit of a primary coded picture are specified in subclause 7.4.1.2.4.

The following constraints shall be obeyed by the order of the coded pictures and non-VCL NAL units within an access unit.

- When an access unit delimiter NAL unit is present, it shall be the first NAL unit. There shall be at most one access unit delimiter NAL unit in any access unit.
- When any SEI NAL units are present, they shall precede the primary coded picture.
- When an SEI NAL unit containing a buffering period SEI message is present, the buffering period SEI message shall be the first SEI message payload of the first SEI NAL unit in the access unit
- The primary coded picture shall precede the corresponding redundant coded pictures.
- When redundant coded pictures are present, they shall be ordered in ascending order of the value of `redundant_pic_cnt`.
- When a sequence parameter set extension NAL unit is present, it shall be the next NAL unit after a sequence parameter set NAL unit having the same value of `seq_parameter_set_id` as in the sequence parameter set extension NAL unit.
- When one or more coded slice of an auxiliary coded picture without partitioning NAL units is present, they shall follow the primary coded picture and all redundant coded pictures (if any).
- When an end of sequence NAL unit is present, it shall follow the primary coded picture and all redundant coded pictures (if any) and all coded slice of an auxiliary coded picture without partitioning NAL units (if any).
- When an end of stream NAL unit is present, it shall be the last NAL unit.
- NAL units having `nal_unit_type` equal to 0, 12, or in the range of 20 to 31, inclusive, shall not precede the first VCL NAL unit of the primary coded picture.

NOTE 1 – Sequence parameter set NAL units or picture parameter set NAL units may be present in an access unit, but cannot follow the last VCL NAL unit of the primary coded picture within the access unit, as this condition would specify the start of a new access unit.

NOTE 2 – When a NAL unit having `nal_unit_type` equal to 7 or 8 is present in an access unit, it may or may not be referred to in the coded pictures of the access unit in which it is present, and may be referred to in coded pictures of subsequent access units.

The structure of access units not containing any NAL units with `nal_unit_type` equal to 0, 7, 8, or in the range of 12 to 18, inclusive, or in the range of 20 to 31, inclusive, is shown in Figure 7-1.

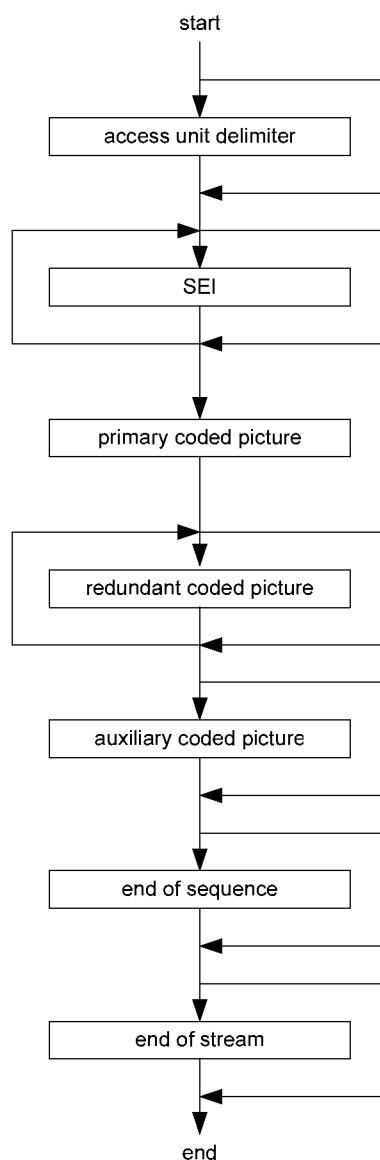


Figure 7-1 – Structure of an access unit not containing any NAL units with nal_unit_type equal to 0, 7, 8, or in the range of 12 to 18, inclusive, or in the range of 20 to 31, inclusive

7.4.1.2.4 Detection of the first VCL NAL unit of a primary coded picture

This subclause specifies constraints on VCL NAL unit syntax that are sufficient to enable the detection of the first VCL NAL unit of each primary coded picture.

Any coded slice NAL unit or coded slice data partition A NAL unit of the primary coded picture of the current access unit shall be different from any coded slice NAL unit or coded slice data partition A NAL unit of the primary coded picture of the previous access unit in one or more of the following ways.

- frame_num differs in value. The value of frame_num used to test this condition is the value of frame_num that appears in the syntax of the slice header, regardless of whether that value is inferred to have been equal to 0 for subsequent use in the decoding process due to the presence of memory_management_control_operation equal to 5.

NOTE 1 – A consequence of the above statement is that a primary coded picture having frame_num equal to 1 cannot contain a memory_management_control_operation equal to 5 unless some other condition listed below is fulfilled for the next primary coded picture that follows after it (if any).

- pic_parameter_set_id differs in value.
- field_pic_flag differs in value.

- `bottom_field_flag` is present in both and differs in value.
- `nal_ref_idc` differs in value with one of the `nal_ref_idc` values being equal to 0.
- `pic_order_cnt_type` is equal to 0 for both and either `pic_order_cnt_lsb` differs in value, or `delta_pic_order_cnt_bottom` differs in value.
- `pic_order_cnt_type` is equal to 1 for both and either `delta_pic_order_cnt[0]` differs in value, or `delta_pic_order_cnt[1]` differs in value.
- `nal_unit_type` differs in value with one of the `nal_unit_type` values being equal to 5.
- `nal_unit_type` is equal to 5 for both and `idr_pic_id` differs in value.

NOTE 2 – Some of the VCL NAL units in redundant coded pictures or some non-VCL NAL units (e.g. an access unit delimiter NAL unit) may also be used for the detection of the boundary between access units, and may therefore aid in the detection of the start of a new primary coded picture.

7.4.1.2.5 Order of VCL NAL units and association to coded pictures

Each VCL NAL unit is part of a coded picture.

The order of the VCL NAL units within a coded IDR picture is constrained as follows.

- If arbitrary slice order is allowed as specified in Annex A, coded slice of an IDR picture NAL units may have any order relative to each other.
- Otherwise (arbitrary slice order is not allowed), the order of coded slice of an IDR picture NAL units shall be in the order of increasing macroblock address for the first macroblock of each coded slice of an IDR picture NAL unit.

The order of the VCL NAL units within a coded non-IDR picture is constrained as follows.

- If arbitrary slice order is allowed as specified in Annex A, coded slice of a non-IDR picture NAL units or coded slice data partition A NAL units may have any order relative to each other. A coded slice data partition A NAL unit with a particular value of `slice_id` shall precede any present coded slice data partition B NAL unit with the same value of `slice_id`. A coded slice data partition A NAL unit with a particular value of `slice_id` shall precede any present coded slice data partition C NAL unit with the same value of `slice_id`. When a coded slice data partition B NAL unit with a particular value of `slice_id` is present, it shall precede any present coded slice data partition C NAL unit with the same value of `slice_id`.
- Otherwise (arbitrary slice order is not allowed), the order of coded slice of a non-IDR picture NAL units or coded slice data partition A NAL units shall be in the order of increasing macroblock address for the first macroblock of each coded slice of a non-IDR picture NAL unit or coded slice data partition A NAL unit. A coded slice data partition A NAL unit with a particular value of `slice_id` shall immediately precede any present coded slice data partition B NAL unit with the same value of `slice_id`. A coded slice data partition A NAL unit with a particular value of `slice_id` shall immediately precede any present coded slice data partition C NAL unit with the same value of `slice_id`, when a coded slice data partition B NAL unit with the same value of `slice_id` is not present. When a coded slice data partition B NAL unit with a particular value of `slice_id` is present, it shall immediately precede any present coded slice data partition C NAL unit with the same value of `slice_id`.

NAL units having `nal_unit_type` equal to 12 may be present in the access unit but shall not precede the first VCL NAL unit of the primary coded picture within the access unit.

NAL units having `nal_unit_type` equal to 0 or in the range of 24 to 31, inclusive, which are unspecified, may be present in the access unit but shall not precede the first VCL NAL unit of the primary coded picture within the access unit.

NAL units having `nal_unit_type` in the range of 20 to 23, inclusive, which are reserved, shall not precede the first VCL NAL unit of the primary coded picture within the access unit (when specified in the future by ITU-T | ISO/IEC).

7.4.2 Raw byte sequence payloads and RBSP trailing bits semantics

7.4.2.1 Sequence parameter set RBSP semantics

profile_idc and **level_idc** indicate the profile and level to which the bitstream conforms, as specified in Annex A.

constraint_set0_flag equal to 1 indicates that the bitstream obeys all constraints specified in subclause A.2.1. **constraint_set0_flag** equal to 0 indicates that the bitstream may or may not obey all constraints specified in subclause A.2.1.

constraint_set1_flag equal to 1 indicates that the bitstream obeys all constraints specified in subclause A.2.2. **constraint_set1_flag** equal to 0 indicates that the bitstream may or may not obey all constraints specified in subclause A.2.2.

constraint_set2_flag equal to 1 indicates that the bitstream obeys all constraints specified in subclause A.2.3. **constraint_set2_flag** equal to 0 indicates that the bitstream may or may not obey all constraints specified in subclause A.2.3.

NOTE 1 – When one or more than one of **constraint_set0_flag**, **constraint_set1_flag**, or **constraint_set2_flag** are equal to 1, the bitstream must obey the constraints of all of the indicated subclauses of subclause A.2. When **profile_idc** is equal to 100, 110, 122, or 144, the values of **constraint_set0_flag**, **constraint_set1_flag**, and **constraint_set2_flag** must all be equal to 0.

constraint_set3_flag indicates the following.

- If **profile_idc** is equal to 66, 77, or 88 and **level_idc** is equal to 11, **constraint_set3_flag** equal to 1 indicates that the bitstream obeys all constraints specified in Annex A for level 1b and **constraint_set3_flag** equal to 0 indicates that the bitstream may or may not obey all constraints specified in Annex A for level 1b.
- Otherwise (**profile_idc** is equal to 100, 110, 122, or 144 or **level_idc** is not equal to 11), the value of 1 for **constraint_set3_flag** is reserved for future use by ITU-T | ISO/IEC. **constraint_set3_flag** shall be equal to 0 in bitstreams conforming to this Recommendation | International Standard when **profile_idc** is equal to 100, 110, 122, or 144 or **level_idc** is not equal to 11. Decoders conforming to this Recommendation | International Standard shall ignore the value of **constraint_set3_flag** when **profile_idc** is equal to 100, 110, 122, or 144 or **level_idc** is not equal to 11.

reserved_zero_4bits shall be equal to 0. Other values of **reserved_zero_4bits** may be specified in the future by ITU-T | ISO/IEC. Decoders shall ignore the value of **reserved_zero_4bits**.

seq_parameter_set_id identifies the sequence parameter set that is referred to by the picture parameter set. The value of **seq_parameter_set_id** shall be in the range of 0 to 31, inclusive.

NOTE 2 – When feasible, encoders should use distinct values of **seq_parameter_set_id** when the values of other sequence parameter set syntax elements differ rather than changing the values of the syntax elements associated with a specific value of **seq_parameter_set_id**.

chroma_format_idc specifies the chroma sampling relative to the luma sampling as specified in subclause 6.2. The value of **chroma_format_idc** shall be in the range of 0 to 3, inclusive. When **chroma_format_idc** is not present, it shall be inferred to be equal to 1 (4:2:0 chroma format).

residual_colour_transform_flag equal to 1 specifies that the residual colour transform is applied as specified in subclause 8.5. **residual_colour_transform_flag** equal to 0 specifies that the residual colour transform is not applied. When **residual_colour_transform_flag** is not present, it shall be inferred to be equal to 0.

bit_depth_luma_minus8 specifies the bit depth of the samples of the luma array and the value of the luma quantisation parameter range offset $QpBdOffset_Y$, as specified by

$$BitDepth_Y = 8 + bit_depth_luma_minus8 \quad (7-1)$$

$$QpBdOffset_Y = 6 * bit_depth_luma_minus8 \quad (7-2)$$

When **bit_depth_luma_minus8** is not present, it shall be inferred to be equal to 0. **bit_depth_luma_minus8** shall be in the range of 0 to 4, inclusive.

bit_depth_chroma_minus8 specifies the bit depth of the samples of the chroma arrays and the value of the chroma quantisation parameter range offset $QpBdOffset_C$, as specified by

$$BitDepth_C = 8 + bit_depth_chroma_minus8 \quad (7-3)$$

$$QpBdOffset_C = 6 * (bit_depth_chroma_minus8 + residual_colour_transform_flag) \quad (7-4)$$

When **bit_depth_chroma_minus8** is not present, it shall be inferred to be equal to 0. **bit_depth_chroma_minus8** shall be in the range of 0 to 4, inclusive.

The variable **RawMbBits** is derived as

$$RawMbBits = 256 * BitDepth_Y + 2 * MbWidthC * MbHeightC * BitDepth_C \quad (7-5)$$

qpprime_y_zero_transform_bypass_flag equal to 1 specifies that, when QP'_Y is equal to 0, a transform bypass operation for the transform coefficient decoding process and picture construction process prior to deblocking filter process as specified in subclause 8.5 shall be applied. **qpprime_y_zero_transform_bypass_flag** equal to 0 specifies that the transform coefficient decoding process and picture construction process prior to deblocking filter process shall not use the transform bypass operation. When **qpprime_y_zero_transform_bypass_flag** is not present, it shall be inferred to be equal to 0.

seq_scaling_matrix_present_flag equal to 1 specifies that the flags **seq_scaling_list_present_flag[i]** for $i = 0..7$ are present. **seq_scaling_matrix_present_flag** equal to 0 specifies that these flags are not present and the sequence-level scaling list specified by **Flat_4x4_16** shall be inferred for $i = 0..5$ and the sequence-level scaling list specified by **Flat_8x8_16** shall be inferred for $i = 6..7$. When **seq_scaling_matrix_present_flag** is not present, it shall be inferred to be equal to 0.

The scaling lists **Flat_4x4_16** and **Flat_8x8_16** are specified as follows:

$$\text{Flat_4x4_16}[i] = 16, \quad \text{with } i = 0..15, \quad (7-6)$$

$$\text{Flat_8x8_16}[i] = 16, \quad \text{with } i = 0..63. \quad (7-7)$$

seq_scaling_list_present_flag[i] equal to 1 specifies that the syntax structure for scaling list i is present in the sequence parameter set. **seq_scaling_list_present_flag[i]** equal to 0 specifies that the syntax structure for scaling list i is not present in the sequence parameter set and the scaling list fall-back rule set A specified in Table 7-2 shall be used to infer the sequence-level scaling list for index i .

Table 7-2 – Assignment of mnemonic names to scaling list indices and specification of fall-back rule

Value of scaling list index	Mnemonic name	Block size	MB prediction type	Component	Scaling list fall-back rule set A	Scaling list fall-back rule set B	Default scaling list
0	SI_4x4_Intra_Y	4x4	Intra	Y	default scaling list	sequence-level scaling list	Default_4x4_Intra
1	SI_4x4_Intra_Cb	4x4	Intra	Cb	scaling list for $i = 0$	scaling list for $i = 0$	Default_4x4_Intra
2	SI_4x4_Intra_Cr	4x4	Intra	Cr	scaling list for $i = 1$	scaling list for $i = 1$	Default_4x4_Intra
3	SI_4x4_Inter_Y	4x4	Inter	Y	default scaling list	sequence-level scaling list	Default_4x4_Inter
4	SI_4x4_Inter_Cb	4x4	Inter	Cb	scaling list for $i = 3$	scaling list for $i = 3$	Default_4x4_Inter
5	SI_4x4_Inter_Cr	4x4	Inter	Cr	scaling list for $i = 4$	scaling list for $i = 4$	Default_4x4_Inter
6	SI_8x8_Intra_Y	8x8	Intra	Y	default scaling list	sequence-level scaling list	Default_8x8_Intra
7	SI_8x8_Inter_Y	8x8	Inter	Y	default scaling list	sequence-level scaling list	Default_8x8_Inter

Table 7-3 specifies the default scaling lists **Default_4x4_Intra** and **Default_4x4_Inter**. Table 7-4 specifies the default scaling lists **Default_8x8_Intra** and **Default_8x8_Inter**.

Table 7-3 – Specification of default scaling lists Default_4x4_Intra and Default_4x4_Inter

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Default_4x4_Intra[idx]	6	13	13	20	20	20	28	28	28	28	32	32	32	37	37	42
Default_4x4_Inter[idx]	10	14	14	20	20	20	24	24	24	24	27	27	27	30	30	34

Table 7-4 – Specification of default scaling lists Default_8x8_Intra and Default_8x8_Inter

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Default_8x8_Intra[idx]	6	10	10	13	11	13	16	16	16	16	18	18	18	18	18	23
Default_8x8_Inter[idx]	9	13	13	15	13	15	17	17	17	17	19	19	19	19	19	21

Table 7-4 (continued) – Specification of default scaling lists Default_8x8_Intra and Default_8x8_Inter

idx	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Default_8x8_Intra[idx]	23	23	23	23	23	25	25	25	25	25	25	25	27	27	27	27
Default_8x8_Inter[idx]	21	21	21	21	21	22	22	22	22	22	22	22	24	24	24	24

Table 7-4 (continued) – Specification of default scaling lists Default_8x8_Intra and Default_8x8_Inter

idx	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
Default_8x8_Intra[idx]	27	27	27	27	29	29	29	29	29	29	29	31	31	31	31	31
Default_8x8_Inter[idx]	24	24	24	24	25	25	25	25	25	25	25	27	27	27	27	27

Table 7-4 (concluded) – Specification of default scaling lists Default_8x8_Intra and Default_8x8_Inter

idx	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Default_8x8_Intra[idx]	31	33	33	33	33	33	36	36	36	36	38	38	38	40	40	42
Default_8x8_Inter[idx]	27	28	28	28	28	28	30	30	30	30	32	32	32	33	33	35

log2_max_frame_num_minus4 specifies the value of the variable MaxFrameNum that is used in frame_num related derivations as follows:

$$\text{MaxFrameNum} = 2^{(\text{log2_max_frame_num_minus4} + 4)} \quad (7-8)$$

The value of log2_max_frame_num_minus4 shall be in the range of 0 to 12, inclusive.

pic_order_cnt_type specifies the method to decode picture order count (as specified in subclause 8.2.1). The value of pic_order_cnt_type shall be in the range of 0 to 2, inclusive.

pic_order_cnt_type shall not be equal to 2 in a coded video sequence that contains any of the following

- an access unit containing a non-reference frame followed immediately by an access unit containing a non-reference picture;
- two access units each containing a field with the two fields together forming a complementary non-reference field pair followed immediately by an access unit containing a non-reference picture;
- an access unit containing a non-reference field followed immediately by an access unit containing another non-reference picture that does not form a complementary non-reference field pair with the first of the two access units.

log2_max_pic_order_cnt_lsb_minus4 specifies the value of the variable `MaxPicOrderCntLsb` that is used in the decoding process for picture order count as specified in subclause 8.2.1 as follows:

$$\text{MaxPicOrderCntLsb} = 2^{(\text{log2_max_pic_order_cnt_lsb_minus4} + 4)} \quad (7-9)$$

The value of `log2_max_pic_order_cnt_lsb_minus4` shall be in the range of 0 to 12, inclusive.

delta_pic_order_always_zero_flag equal to 1 specifies that `delta_pic_order_cnt[0]` and `delta_pic_order_cnt[1]` are not present in the slice headers of the sequence and shall be inferred to be equal to 0. `delta_pic_order_always_zero_flag` equal to 0 specifies that `delta_pic_order_cnt[0]` is present in the slice headers of the sequence and `delta_pic_order_cnt[1]` may be present in the slice headers of the sequence.

offset_for_non_ref_pic is used to calculate the picture order count of a non-reference picture as specified in 8.2.1. The value of `offset_for_non_ref_pic` shall be in the range of -2^{31} to $2^{31} - 1$, inclusive.

offset_for_top_to_bottom_field is used to calculate the picture order count of a bottom field as specified in subclause 8.2.1. The value of `offset_for_top_to_bottom_field` shall be in the range of -2^{31} to $2^{31} - 1$, inclusive.

num_ref_frames_in_pic_order_cnt_cycle is used in the decoding process for picture order count as specified in subclause 8.2.1. The value of `num_ref_frames_in_pic_order_cnt_cycle` shall be in the range of 0 to 255, inclusive.

offset_for_ref_frame[i] is an element of a list of `num_ref_frames_in_pic_order_cnt_cycle` values used in the decoding process for picture order count as specified in subclause 8.2.1. The value of `offset_for_ref_frame[i]` shall be in the range of -2^{31} to $2^{31} - 1$, inclusive.

num_ref_frames specifies the maximum number of short-term and long-term reference frames, complementary reference field pairs, and non-paired reference fields that may be used by the decoding process for inter prediction of any picture in the sequence. `num_ref_frames` also determines the size of the sliding window operation as specified in subclause 8.2.5.3. The value of `num_ref_frames` shall be in the range of 0 to `MaxDpbSize` (as specified in subclause A.3.1 or A.3.2), inclusive.

gaps_in_frame_num_value_allowed_flag specifies the allowed values of `frame_num` as specified in subclause 7.4.3 and the decoding process in case of an inferred gap between values of `frame_num` as specified in subclause 8.2.5.2.

pic_width_in_mbs_minus1 plus 1 specifies the width of each decoded picture in units of macroblocks.

The variable for the picture width in units of macroblocks is derived as follows

$$\text{PicWidthInMbs} = \text{pic_width_in_mbs_minus1} + 1 \quad (7-10)$$

The variable for picture width for the luma component is derived as follows

$$\text{PicWidthInSamples}_L = \text{PicWidthInMbs} * 16 \quad (7-11)$$

The variable for picture width for the chroma components is derived as follows

$$\text{PicWidthInSamples}_C = \text{PicWidthInMbs} * \text{MbWidthC} \quad (7-12)$$

pic_height_in_map_units_minus1 plus 1 specifies the height in slice group map units of a decoded frame or field.

The variables `PicHeightInMapUnits` and `PicSizeInMapUnits` are derived as follows

$$\text{PicHeightInMapUnits} = \text{pic_height_in_map_units_minus1} + 1 \quad (7-13)$$

$$\text{PicSizeInMapUnits} = \text{PicWidthInMbs} * \text{PicHeightInMapUnits} \quad (7-14)$$

frame_mbs_only_flag equal to 0 specifies that coded pictures of the coded video sequence may either be coded fields or coded frames. `frame_mbs_only_flag` equal to 1 specifies that every coded picture of the coded video sequence is a coded frame containing only frame macroblocks.

The allowed range of values for `pic_width_in_mbs_minus1`, `pic_height_in_map_units_minus1`, and `frame_mbs_only_flag` is specified by constraints in Annex A.

Depending on `frame_mbs_only_flag`, semantics are assigned to `pic_height_in_map_units_minus1` as follows.

- If `frame_mbs_only_flag` is equal to 0, `pic_height_in_map_units_minus1` plus 1 is the height of a field in units of macroblocks.
- Otherwise (`frame_mbs_only_flag` is equal to 1), `pic_height_in_map_units_minus1` plus 1 is the height of a frame in units of macroblocks.

The variable `FrameHeightInMbs` is derived as follows

$$\text{FrameHeightInMbs} = (2 - \text{frame_mbs_only_flag}) * \text{PicHeightInMapUnits} \quad (7-15)$$

`mb_adaptive_frame_field_flag` equal to 0 specifies no switching between frame and field macroblocks within a picture. `mb_adaptive_frame_field_flag` equal to 1 specifies the possible use of switching between frame and field macroblocks within frames. When `mb_adaptive_frame_field_flag` is not present, it shall be inferred to be equal to 0.

`direct_8x8_inference_flag` specifies the method used in the derivation process for luma motion vectors for `B_Skip`, `B_Direct_16x16` and `B_Direct_8x8` as specified in subclause 8.4.1.2. When `frame_mbs_only_flag` is equal to 0, `direct_8x8_inference_flag` shall be equal to 1.

`frame_cropping_flag` equal to 1 specifies that the frame cropping offset parameters follow next in the sequence parameter set. `frame_cropping_flag` equal to 0 specifies that the frame cropping offset parameters are not present.

`frame_crop_left_offset`, `frame_crop_right_offset`, `frame_crop_top_offset`, `frame_crop_bottom_offset` specify the samples of the pictures in the coded video sequence that are output from the decoding process, in terms of a rectangular region specified in frame coordinates for output.

The variables `CropUnitX` and `CropUnitY` are derived as follows:

- If `chroma_format_idc` is equal to 0, `CropUnitX` and `CropUnitY` are derived as

$$\text{CropUnitX} = 1 \quad (7-16)$$

$$\text{CropUnitY} = 2 - \text{frame_mbs_only_flag} \quad (7-17)$$

- Otherwise (`chroma_format_idc` is equal to 1, 2, or 3), `CropUnitX` and `CropUnitY` are derived as

$$\text{CropUnitX} = \text{SubWidthC} \quad (7-18)$$

$$\text{CropUnitY} = \text{SubHeightC} * (2 - \text{frame_mbs_only_flag}) \quad (7-19)$$

The frame cropping rectangle contains luma samples with horizontal frame coordinates from `CropUnitX * frame_crop_left_offset` to `PicWidthInSamplesL - (CropUnitX * frame_crop_right_offset + 1)` and vertical frame coordinates from `CropUnitY * frame_crop_top_offset` to `(16 * FrameHeightInMbs) - (CropUnitY * frame_crop_bottom_offset + 1)`, inclusive. The value of `frame_crop_left_offset` shall be in the range of 0 to `(PicWidthInSamplesL / CropUnitX) - (frame_crop_right_offset + 1)`, inclusive; and the value of `frame_crop_top_offset` shall be in the range of 0 to `(16 * FrameHeightInMbs / CropUnitY) - (frame_crop_bottom_offset + 1)`, inclusive.

When `frame_cropping_flag` is equal to 0, the values of `frame_crop_left_offset`, `frame_crop_right_offset`, `frame_crop_top_offset`, and `frame_crop_bottom_offset` shall be inferred to be equal to 0.

When `chroma_format_idc` is not equal to 0, the corresponding specified samples of the two chroma arrays are the samples having frame coordinates `(x / SubWidthC, y / SubHeightC)`, where `(x, y)` are the frame coordinates of the specified luma samples.

For decoded fields, the specified samples of the decoded field are the samples that fall within the rectangle specified in frame coordinates.

`vui_parameters_present_flag` equal to 1 specifies that the `vui_parameters()` syntax structure as specified in Annex E is present. `vui_parameters_present_flag` equal to 0 specifies that the `vui_parameters()` syntax structure as specified in Annex E is not present.

7.4.2.1.1 Scaling list semantics

`delta_scale` is used to derive the `j`-th element of the scaling list for `j` in the range of 0 to `sizeOfScalingList - 1`, inclusive. The value of `delta_scale` shall be in the range of -128 to +127, inclusive.

When `useDefaultScalingMatrixFlag` is derived to be equal to 1, the scaling list shall be inferred to be equal to the default scaling list as specified in Table 7-2.

7.4.2.1.2 Sequence parameter set extension RBSP semantics

seq_parameter_set_id identifies the sequence parameter set associated with the sequence parameter set extension. The value of seq_parameter_set_id shall be in the range of 0 to 31, inclusive.

aux_format_idc equal to 0 indicates that there are no auxiliary coded pictures in the coded video sequence. aux_format_idc equal to 1 indicates that exactly one auxiliary coded picture is present in each access unit of the coded video sequence, and that for alpha blending purposes the decoded samples of the associated primary coded picture in each access unit should be multiplied by the interpretation sample values of the auxiliary coded picture in the access unit in the display process after output from the decoding process. aux_format_idc equal to 2 indicates that exactly one auxiliary coded picture exists in each access unit of the coded video sequence, and that for alpha blending purposes the decoded samples of the associated primary coded picture in each access unit should not be multiplied by the interpretation sample values of the auxiliary coded picture in the access unit in the display process after output from the decoding process. aux_format_idc equal to 3 indicates that exactly one auxiliary coded picture exists in each access unit of the coded video sequence, and that the usage of the auxiliary coded pictures is unspecified. The value of aux_format_idc shall be in the range of 0 to 3, inclusive. Values greater than 3 for aux_format_idc are reserved to indicate the presence of exactly one auxiliary coded picture in each access unit of the coded video sequence for purposes to be specified in the future by ITU-T | ISO/IEC. When aux_format_idc is not present, it shall be inferred to be equal to 0.

NOTE 1 – Decoders conforming to this Recommendation | International Standard are not required to decode auxiliary coded pictures.

bit_depth_aux_minus8 specifies the bit depth of the samples of the sample array of the auxiliary coded picture. bit_depth_aux_minus8 shall be in the range of 0 to 4, inclusive.

alpha_incr_flag equal to 0 indicates that the interpretation sample value for each decoded auxiliary coded picture sample value is equal to the decoded auxiliary coded picture sample value for purposes of alpha blending. alpha_incr_flag equal to 1 indicates that, for purposes of alpha blending, after decoding the auxiliary coded picture samples, any auxiliary coded picture sample value that is greater than $\text{Min}(\text{alpha_opaque_value}, \text{alpha_transparent_value})$ should be increased by one to obtain the interpretation sample value for the auxiliary coded picture sample, and any auxiliary coded picture sample value that is less than or equal to $\text{Min}(\text{alpha_opaque_value}, \text{alpha_transparent_value})$ should be used without alteration as the interpretation sample value for the decoded auxiliary coded picture sample value.

alpha_opaque_value specifies the interpretation sample value of an auxiliary coded picture sample for which the associated luma and chroma samples of the same access unit are considered opaque for purposes of alpha blending. The number of bits used for the representation of the alpha_opaque_value syntax element is bit_depth_aux_minus8 + 9 bits.

alpha_transparent_value specifies the interpretation sample value of an auxiliary coded picture sample for which the associated luma and chroma samples of the same access unit are considered transparent for purposes of alpha blending. The number of bits used for the representation of the alpha_transparent_value syntax element is bit_depth_aux_minus8 + 9 bits.

When alpha_incr_flag is equal to 1, alpha_transparent_value shall not be equal to alpha_opaque_value and $\text{Log2}(\text{Abs}(\text{alpha_opaque_value} - \text{alpha_transparent_value}))$ shall have an integer value. A value of alpha_transparent_value that is equal to alpha_opaque_value indicates that the auxiliary coded picture is not intended for alpha blending purposes.

NOTE 2 – For alpha blending purposes, alpha_opaque_value may be greater than alpha_transparent_value, or it may be less than alpha_transparent_value. Interpretation sample values should be clipped to the range of alpha_opaque_value to alpha_transparent_value, inclusive.

The decoding of the sequence parameter set extension and the decoding of auxiliary coded pictures is not required for conformance with this Recommendation | International Standard.

The syntax of each coded slice of an auxiliary coded picture shall obey the same constraints as a coded slice of a redundant picture, with the following differences of constraints.

- The following applies in regard to whether the primary coded picture is an IDR picture.
 - If the primary coded picture is an IDR picture, the auxiliary coded slice syntax shall correspond to that of a slice having nal_unit_type equal to 5 (a slice of an IDR picture);
 - Otherwise (the primary coded picture is not an IDR picture), the auxiliary coded slice syntax shall correspond to that of a slice having nal_unit_type equal to 1 (a slice of a non-IDR picture).
- The slices of an auxiliary coded picture (when present) shall contain all macroblocks corresponding to those of the primary coded picture.
- redundant_pic_cnt shall be equal to 0 in all auxiliary coded slices.

The (optional) decoding process for the decoding of auxiliary coded pictures is the same as if the auxiliary coded pictures were primary coded pictures in a separate coded video stream that differs from the primary coded pictures in the current coded video stream in the following ways.

- The IDR or non-IDR status of each auxiliary coded picture shall be inferred to be the same as the IDR or non-IDR status of the primary picture in the same access unit, rather than being inferred from the value of `nal_ref_idc`.
- The value of `chroma_format_idc` shall be inferred to be equal to 0 for the decoding of the auxiliary coded pictures.
- The value of `bit_depth_luma_minus8` shall be inferred to be equal to `bit_depth_aux_minus8` for the decoding of the auxiliary coded pictures.

NOTE 3 – Alpha blending composition is normally performed with a background picture B, a foreground picture F, and a decoded auxiliary coded picture A, all of the same size. Assume for purposes of example illustration that the chroma resolution of B and F have been upsampled to the same resolution as the luma. Denote corresponding samples of B, F and A by b , f and a , respectively. Denote luma and chroma samples by subscripts Y, Cb and Cr.

Define the variables `alphaRange`, `alphaFwt` and `alphaBwt` as follows:

$$\text{alphaRange} = \text{Abs}(\text{alpha_opaque_value} - \text{alpha_transparent_value})$$

$$\text{alphaFwt} = \text{Abs}(a - \text{alpha_transparent_value})$$

$$\text{alphaBwt} = \text{Abs}(a - \text{alpha_opaque_value})$$

Then, in alpha blending composition, samples d of the displayed picture D may be calculated as

$$d_Y = (\text{alphaFwt} * f_Y + \text{alphaBwt} * b_Y + \text{alphaRange}/2) / \text{alphaRange}$$

$$d_{CB} = (\text{alphaFwt} * f_{CB} + \text{alphaBwt} * b_{CB} + \text{alphaRange}/2) / \text{alphaRange}$$

$$d_{CR} = (\text{alphaFwt} * f_{CR} + \text{alphaBwt} * b_{CR} + \text{alphaRange}/2) / \text{alphaRange}$$

The samples of pictures D, F and B could also represent red, green, and blue component values (see subclause E.2.1). Here we have assumed Y, Cb and Cr component values. Each component, e.g. Y, is assumed for purposes of example illustration above to have the same bit depth in each of the pictures D, F and B. However, different components, e.g. Y and Cb, need not have the same bit depth in this example.

When `aux_format_idc` is equal to 1, F would be the decoded picture obtained from the decoded luma and chroma, and A would be the decoded picture obtained from the decoded auxiliary coded picture. In this case, the indicated example alpha blending composition involves multiplying the samples of F by factors obtained from the samples of A.

A picture format that is useful for editing or direct viewing, and that is commonly used, is called pre-multiplied-black video. If the foreground picture was F, then the pre-multiplied-black video S is given by

$$s_Y = (\text{alphaFwt} * f_Y) / \text{alphaRange}$$

$$s_{CB} = (\text{alphaFwt} * f_{CB}) / \text{alphaRange}$$

$$s_{CR} = (\text{alphaFwt} * f_{CR}) / \text{alphaRange}$$

Pre-multiplied-black video has the characteristic that the picture S will appear correct if displayed against a black background. For a non-black background B, the composition of the displayed picture D may be calculated as

$$d_Y = s_Y + (\text{alphaBwt} * b_Y + \text{alphaRange}/2) / \text{alphaRange}$$

$$d_{CB} = s_{CB} + (\text{alphaBwt} * b_{CB} + \text{alphaRange}/2) / \text{alphaRange}$$

$$d_{CR} = s_{CR} + (\text{alphaBwt} * b_{CR} + \text{alphaRange}/2) / \text{alphaRange}$$

When `aux_format_idc` is equal to 2, S would be the decoded picture obtained from the decoded luma and chroma, and A would again be the decoded picture obtained from the decoded auxiliary coded picture. In this case, alpha blending composition does not involve multiplication of the samples of S by factors obtained from the samples of A.

additional_extension_flag equal to 0 indicates that no additional data follows within the sequence parameter set extension syntax structure prior to the Rbsp trailing bits. The value of `additional_extension_flag` shall be equal to 0. The value of 1 for `additional_extension_flag` is reserved for future use by ITU-T | ISO/IEC. Decoders that conform to this Recommendation | International Standard shall ignore all data that follows the value of 1 for `additional_extension_flag` in a sequence parameter set extension NAL unit.

7.4.2.2 Picture parameter set Rbsp semantics

pic_parameter_set_id identifies the picture parameter set that is referred to in the slice header. The value of `pic_parameter_set_id` shall be in the range of 0 to 255, inclusive.

seq_parameter_set_id refers to the active sequence parameter set. The value of `seq_parameter_set_id` shall be in the range of 0 to 31, inclusive.

entropy_coding_mode_flag selects the entropy decoding method to be applied for the syntax elements for which two descriptors appear in the syntax tables as follows.

- If `entropy_coding_mode_flag` is equal to 0, the method specified by the left descriptor in the syntax table is applied (Exp-Golomb coded, see subclause 9.1 or CAVLC, see subclause 9.2).
- Otherwise (`entropy_coding_mode_flag` is equal to 1), the method specified by the right descriptor in the syntax table is applied (CABAC, see subclause 9.3).

pic_order_present_flag equal to 1 specifies that the picture order count related syntax elements are present in the slice headers as specified in subclause 7.3.3. **pic_order_present_flag** equal to 0 specifies that the picture order count related syntax elements are not present in the slice headers.

num_slice_groups_minus1 plus 1 specifies the number of slice groups for a picture. When **num_slice_groups_minus1** is equal to 0, all slices of the picture belong to the same slice group. The allowed range of **num_slice_groups_minus1** is specified in Annex A.

slice_group_map_type specifies how the mapping of slice group map units to slice groups is coded. The value of **slice_group_map_type** shall be in the range of 0 to 6, inclusive.

slice_group_map_type equal to 0 specifies interleaved slice groups.

slice_group_map_type equal to 1 specifies a dispersed slice group mapping.

slice_group_map_type equal to 2 specifies one or more “foreground” slice groups and a “leftover” slice group.

slice_group_map_type values equal to 3, 4, and 5 specify changing slice groups. When **num_slice_groups_minus1** is not equal to 1, **slice_group_map_type** shall not be equal to 3, 4, or 5.

slice_group_map_type equal to 6 specifies an explicit assignment of a slice group to each slice group map unit.

Slice group map units are specified as follows.

- If **frame_mbs_only_flag** is equal to 0 and **mb_adaptive_frame_field_flag** is equal to 1 and the coded picture is a frame, the slice group map units are macroblock pair units.
- Otherwise, if **frame_mbs_only_flag** is equal to 1 or a coded picture is a field, the slice group map units are units of macroblocks.
- Otherwise (**frame_mbs_only_flag** is equal to 0 and **mb_adaptive_frame_field_flag** is equal to 0 and the coded picture is a frame), the slice group map units are units of two macroblocks that are vertically contiguous as in a frame macroblock pair of an MBAFF frame.

run_length_minus1[i] is used to specify the number of consecutive slice group map units to be assigned to the *i*-th slice group in raster scan order of slice group map units. The value of **run_length_minus1[i]** shall be in the range of 0 to **PicSizeInMapUnits** - 1, inclusive.

top_left[i] and **bottom_right[i]** specify the top-left and bottom-right corners of a rectangle, respectively. **top_left[i]** and **bottom_right[i]** are slice group map unit positions in a raster scan of the picture for the slice group map units. For each rectangle *i*, all of the following constraints shall be obeyed by the values of the syntax elements **top_left[i]** and **bottom_right[i]**

- **top_left[i]** shall be less than or equal to **bottom_right[i]** and **bottom_right[i]** shall be less than **PicSizeInMapUnits**.
- (**top_left[i]** % **PicWidthInMbs**) shall be less than or equal to the value of (**bottom_right[i]** % **PicWidthInMbs**).

slice_group_change_direction_flag is used with **slice_group_map_type** to specify the refined map type when **slice_group_map_type** is 3, 4, or 5.

slice_group_change_rate_minus1 is used to specify the variable **SliceGroupChangeRate**. **SliceGroupChangeRate** specifies the multiple in number of slice group map units by which the size of a slice group can change from one picture to the next. The value of **slice_group_change_rate_minus1** shall be in the range of 0 to **PicSizeInMapUnits** - 1, inclusive. The **SliceGroupChangeRate** variable is specified as follows:

$$\text{SliceGroupChangeRate} = \text{slice_group_change_rate_minus1} + 1 \quad (7-20)$$

pic_size_in_map_units_minus1 is used to specify the number of slice group map units in the picture. **pic_size_in_map_units_minus1** shall be equal to **PicSizeInMapUnits** - 1.

slice_group_id[i] identifies a slice group of the *i*-th slice group map unit in raster scan order. The size of the **slice_group_id[i]** syntax element is $\text{Ceil}(\text{Log2}(\text{num_slice_groups_minus1} + 1))$ bits. The value of **slice_group_id[i]** shall be in the range of 0 to **num_slice_groups_minus1**, inclusive.

num_ref_idx_l0_active_minus1 specifies the maximum reference index for reference picture list 0 that shall be used to decode each slice of the picture in which list 0 prediction is used when **num_ref_idx_active_override_flag** is equal to 0 for the slice. When **MbaffFrameFlag** is equal to 1, **num_ref_idx_l0_active_minus1** is the maximum index value for the decoding of frame macroblocks and $2 * \text{num_ref_idx_l0_active_minus1} + 1$ is the maximum index value for the decoding of field macroblocks. The value of **num_ref_idx_l0_active_minus1** shall be in the range of 0 to 31, inclusive.

num_ref_idx_l1_active_minus1 has the same semantics as num_ref_idx_l0_active_minus1 with l0 and list 0 replaced by l1 and list 1, respectively.

weighted_pred_flag equal to 0 specifies that weighted prediction shall not be applied to P and SP slices. weighted_pred_flag equal to 1 specifies that weighted prediction shall be applied to P and SP slices.

weighted_bipred_idc equal to 0 specifies that the default weighted prediction shall be applied to B slices. weighted_bipred_idc equal to 1 specifies that explicit weighted prediction shall be applied to B slices. weighted_bipred_idc equal to 2 specifies that implicit weighted prediction shall be applied to B slices. The value of weighted_bipred_idc shall be in the range of 0 to 2, inclusive.

pic_init_qp_minus26 specifies the initial value minus 26 of SliceQP_Y for each slice. The initial value is modified at the slice layer when a non-zero value of slice_qp_delta is decoded, and is modified further when a non-zero value of mb_qp_delta is decoded at the macroblock layer. The value of pic_init_qp_minus26 shall be in the range of $-(26 + QpBdOffset_Y)$ to +25, inclusive.

pic_init_qs_minus26 specifies the initial value minus 26 of SliceQS_Y for all macroblocks in SP or SI slices. The initial value is modified at the slice layer when a non-zero value of slice_qs_delta is decoded. The value of pic_init_qs_minus26 shall be in the range of -26 to +25, inclusive.

chroma_qp_index_offset specifies the offset that shall be added to QP_Y and QS_Y for addressing the table of QP_C values for the Cb chroma component. The value of chroma_qp_index_offset shall be in the range of -12 to +12, inclusive.

deblocking_filter_control_present_flag equal to 1 specifies that a set of syntax elements controlling the characteristics of the deblocking filter is present in the slice header. deblocking_filter_control_present_flag equal to 0 specifies that the set of syntax elements controlling the characteristics of the deblocking filter is not present in the slice headers and their inferred values are in effect.

constrained_intra_pred_flag equal to 0 specifies that intra prediction allows usage of residual data and decoded samples of neighbouring macroblocks coded using Inter macroblock prediction modes for the prediction of macroblocks coded using Intra macroblock prediction modes. constrained_intra_pred_flag equal to 1 specifies constrained intra prediction, in which case prediction of macroblocks coded using Intra macroblock prediction modes only uses residual data and decoded samples from I or SI macroblock types.

redundant_pic_cnt_present_flag equal to 0 specifies that the redundant_pic_cnt syntax element is not present in slice headers, data partitions B, and data partitions C that refer (either directly or by association with a corresponding data partition A) to the picture parameter set. redundant_pic_cnt_present_flag equal to 1 specifies that the redundant_pic_cnt syntax element is present in all slice headers, data partitions B, and data partitions C that refer (either directly or by association with a corresponding data partition A) to the picture parameter set.

transform_8x8_mode_flag equal to 1 specifies that the 8x8 transform decoding process may be in use (see subclause 8.5). transform_8x8_mode_flag equal to 0 specifies that the 8x8 transform decoding process is not in use. When transform_8x8_mode_flag is not present, it shall be inferred to be 0.

pic_scaling_matrix_present_flag equal to 1 specifies that parameters are present to modify the scaling lists specified in the sequence parameter set. pic_scaling_matrix_present_flag equal to 0 specifies that the scaling lists used for the picture shall be inferred to be equal to those specified by the sequence parameter set. When pic_scaling_matrix_present_flag is not present, it shall be inferred to be equal to 0.

pic_scaling_list_present_flag[i] equal to 1 specifies that the scaling list syntax structure is present to specify the scaling list for index i. pic_scaling_list_present_flag[i] equal to 0 specifies that the syntax structure for scaling list i is not present in the picture parameter set and that depending on the value of seq_scaling_matrix_present_flag, the following applies.

- If seq_scaling_matrix_present_flag is equal to 0, the scaling list fall-back rule set A as specified in Table 7-2 shall be used to derive the picture-level scaling list for index i.
- Otherwise (seq_scaling_matrix_present_flag is equal to 1), the scaling list fall-back rule set B as specified in Table 7-2 shall be used to derive the picture-level scaling list for index i.

second_chroma_qp_index_offset specifies the offset that shall be added to QP_Y and QS_Y for addressing the table of QP_C values for the Cr chroma component. The value of second_chroma_qp_index_offset shall be in the range of -12 to +12, inclusive.

When second_chroma_qp_index_offset is not present, it shall be inferred to be equal to chroma_qp_index_offset.

7.4.2.3 Supplemental enhancement information RBSP semantics

Supplemental Enhancement Information (SEI) contains information that is not necessary to decode the samples of coded pictures from VCL NAL units.

7.4.2.3.1 Supplemental enhancement information message semantics

An SEI NAL unit contains one or more SEI messages. Each SEI message consists of the variables specifying the type payloadType and size payloadSize of the SEI payload. SEI payloads are specified in Annex D. The derived SEI payload size payloadSize is specified in bytes and shall be equal to the number of bytes in the SEI payload.

ff_byte is a byte equal to 0xFF identifying a need for a longer representation of the syntax structure that it is used within.

last_payload_type_byte is the last byte of the payload type of an SEI message.

last_payload_size_byte is the last byte of the size of an SEI message.

7.4.2.4 Access unit delimiter RBSP semantics

The access unit delimiter may be used to indicate the type of slices present in a primary coded picture and to simplify the detection of the boundary between access units. There is no normative decoding process associated with the access unit delimiter.

primary_pic_type indicates that the slice_type values for all slices of the primary coded picture are members of the set listed in Table 7-5 for the given value of primary_pic_type.

Table 7-5 – Meaning of primary_pic_type

primary_pic_type	slice_type values that may be present in the primary coded picture
0	I
1	I, P
2	I, P, B
3	SI
4	SI, SP
5	I, SI
6	I, SI, P, SP
7	I, SI, P, SP, B

7.4.2.5 End of sequence RBSP semantics

The end of sequence RBSP specifies that the next subsequent access unit in the bitstream in decoding order (if any) shall be an IDR access unit. The syntax content of the SODB and RBSP for the end of sequence RBSP are empty. No normative decoding process is specified for an end of sequence RBSP.

7.4.2.6 End of stream RBSP semantics

The end of stream RBSP indicates that no additional NAL units shall be present in the bitstream that are subsequent to the end of stream RBSP in decoding order. The syntax content of the SODB and RBSP for the end of stream RBSP are empty. No normative decoding process is specified for an end of stream RBSP.

7.4.2.7 Filler data RBSP semantics

The filler data RBSP contains bytes whose value shall be equal to 0xFF. No normative decoding process is specified for a filler data RBSP.

ff_byte is a byte equal to 0xFF.

7.4.2.8 Slice layer without partitioning RBSP semantics

The slice layer without partitioning RBSP consists of a slice header and slice data.

7.4.2.9 Slice data partition RBSP semantics

7.4.2.9.1 Slice data partition A RBSP semantics

When slice data partitioning is in use, the coded data for a single slice is divided into three separate partitions. Partition A contains all syntax elements of category 2.

Category 2 syntax elements include all syntax elements in the slice header and slice data syntax structures other than the syntax elements in the residual() syntax structure.

slice_id identifies the slice associated with the data partition. Each slice shall have a unique slice_id value within the coded picture that contains the slice. When arbitrary slice order is not allowed as specified in Annex A, the first slice of a coded picture, in decoding order, shall have slice_id equal to 0 and the value of slice_id shall be incremented by one for each subsequent slice of the coded picture in decoding order.

The range of slice_id is specified as follows.

- If MbaffFrameFlag is equal to 0, slice_id shall be in the range of 0 to PicSizeInMbs - 1, inclusive.
- Otherwise (MbaffFrameFlag is equal to 1), slice_id shall be in the range of 0 to PicSizeInMbs / 2 - 1, inclusive.

7.4.2.9.2 Slice data partition B RBSP semantics

When slice data partitioning is in use, the coded data for a single slice is divided into one to three separate partitions. Slice data partition B contains all syntax elements of category 3.

Category 3 syntax elements include all syntax elements in the residual() syntax structure and in syntax structures used within that syntax structure for collective macroblock types I and SI as specified in Table 7-10.

slice_id has the same semantics as specified in subclause 7.4.2.9.1.

redundant_pic_cnt shall be equal to 0 for slices and slice data partitions belonging to the primary coded picture. The redundant_pic_cnt shall be greater than 0 for coded slices and coded slice data partitions in redundant coded pictures. When redundant_pic_cnt is not present, its value shall be inferred to be equal to 0. The value of redundant_pic_cnt shall be in the range of 0 to 127, inclusive.

The presence of a slice data partition B RBSP is specified as follows.

- If the syntax elements of a slice data partition A RBSP indicate the presence of any syntax elements of category 3 in the slice data for a slice, a slice data partition B RBSP shall be present having the same value of slice_id and redundant_pic_cnt as in the slice data partition A RBSP.
- Otherwise (the syntax elements of a slice data partition A RBSP do not indicate the presence of any syntax elements of category 3 in the slice data for a slice), no slice data partition B RBSP shall be present having the same value of slice_id and redundant_pic_cnt as in the slice data partition A RBSP.

7.4.2.9.3 Slice data partition C RBSP semantics

When slice data partitioning is in use, the coded data for a single slice is divided into three separate partitions. Slice data partition C contains all syntax elements of category 4.

Category 4 syntax elements include all syntax elements in the residual() syntax structure and in syntax structures used within that syntax structure for collective macroblock types P and B as specified in Table 7-10.

slice_id has the same semantics as specified in subclause 7.4.2.9.1.

redundant_pic_cnt has the same semantics as specified in subclause 7.4.2.9.2.

The presence of a slice data partition C RBSP is specified as follows.

- If the syntax elements of a slice data partition A RBSP indicate the presence of any syntax elements of category 4 in the slice data for a slice, a slice data partition C RBSP shall be present having the same value of slice_id and redundant_pic_cnt as in the slice data partition A RBSP.
- Otherwise (the syntax elements of a slice data partition A RBSP do not indicate the presence of any syntax elements of category 4 in the slice data for a slice), no slice data partition C RBSP shall be present having the same value of slice_id and redundant_pic_cnt as in the slice data partition A RBSP.

7.4.2.10 RBSP slice trailing bits semantics

cabac_zero_word is a byte-aligned sequence of two bytes equal to 0x0000.

Let NumBytesInVclNALunits be the sum of the values of NumBytesInNALunit for all VCL NAL units of a coded picture.

Let BinCountsInNALunits be the number of times that the parsing process function DecodeBin(), specified in subclause 9.3.3.2, is invoked to decode the contents of all VCL NAL units of a coded picture. When entropy_coding_mode_flag is equal to 1, BinCountsInNALunits shall not exceed $(32 \div 3) * \text{NumBytesInVclNALunits} + (\text{RawMbBits} * \text{PicSizeInMbs}) \div 32$.

NOTE – The constraint on the maximum number of bins resulting from decoding the contents of the slice layer NAL units can be met by inserting a number of cabac_zero_word syntax elements to increase the value of NumBytesInVclNALunits. Each cabac_zero_word is represented in a NAL unit by the three-byte sequence 0x000003 (as a result of the constraints on NAL unit contents that result in requiring inclusion of an emulation_prevention_three_byte for each cabac_zero_word).

7.4.2.11 RBSP trailing bits semantics

rbsp_stop_one_bit shall be equal to 1.

rbsp_alignment_zero_bit shall be equal to 0.

7.4.3 Slice header semantics

When present, the value of the slice header syntax elements pic_parameter_set_id, frame_num, field_pic_flag, bottom_field_flag, idr_pic_id, pic_order_cnt_lsb, delta_pic_order_cnt_bottom, delta_pic_order_cnt[0], delta_pic_order_cnt[1], sp_for_switch_flag, and slice_group_change_cycle shall be the same in all slice headers of a coded picture.

first_mb_in_slice specifies the address of the first macroblock in the slice. When arbitrary slice order is not allowed as specified in Annex A, the value of first_mb_in_slice shall not be less than the value of first_mb_in_slice for any other slice of the current picture that precedes the current slice in decoding order.

The first macroblock address of the slice is derived as follows.

- If MbaffFrameFlag is equal to 0, first_mb_in_slice is the macroblock address of the first macroblock in the slice, and first_mb_in_slice shall be in the range of 0 to PicSizeInMbs - 1, inclusive.
- Otherwise (MbaffFrameFlag is equal to 1), first_mb_in_slice * 2 is the macroblock address of the first macroblock in the slice, which is the top macroblock of the first macroblock pair in the slice, and first_mb_in_slice shall be in the range of 0 to PicSizeInMbs / 2 - 1, inclusive.

slice_type specifies the coding type of the slice according to Table 7-6.

Table 7-6 – Name association to slice_type

slice_type	Name of slice_type
0	P (P slice)
1	B (B slice)
2	I (I slice)
3	SP (SP slice)
4	SI (SI slice)
5	P (P slice)
6	B (B slice)
7	I (I slice)
8	SP (SP slice)
9	SI (SI slice)

slice_type values in the range 5..9 specify, in addition to the coding type of the current slice, that all other slices of the current coded picture shall have a value of slice_type equal to the current value of slice_type or equal to the current value of slice_type - 5.

When nal_unit_type is equal to 5 (IDR picture), slice_type shall be equal to 2, 4, 7, or 9.

When num_ref_frames is equal to 0, slice_type shall be equal to 2, 4, 7, or 9.

pic_parameter_set_id specifies the picture parameter set in use. The value of pic_parameter_set_id shall be in the range of 0 to 255, inclusive.

frame_num is used as an identifier for pictures and shall be represented by $\log_2 \text{max_frame_num_minus4} + 4$ bits in the bitstream. **frame_num** is constrained as follows:

The variable **PrevRefFrameNum** is derived as follows.

- If the current picture is an IDR picture, **PrevRefFrameNum** is set equal to 0.
- Otherwise (the current picture is not an IDR picture), **PrevRefFrameNum** is set as follows.
 - If the decoding process for gaps in **frame_num** specified in subclause 8.2.5.2 was invoked by the decoding process for an access unit that contained a non-reference picture that followed the previous access unit in decoding order that contained a reference picture, **PrevRefFrameNum** is set equal to the value of **frame_num** for the last of the "non-existing" reference frames inferred by the decoding process for gaps in **frame_num** specified in subclause 8.2.5.2.
 - Otherwise, **PrevRefFrameNum** is set equal to the value of **frame_num** for the previous access unit in decoding order that contained a reference picture.

The value of **frame_num** is constrained as follows.

- If the current picture is an IDR picture, **frame_num** shall be equal to 0.
- Otherwise (the current picture is not an IDR picture), referring to the primary coded picture in the previous access unit in decoding order that contains a reference picture as the preceding reference picture, the value of **frame_num** for the current picture shall not be equal to **PrevRefFrameNum** unless all of the following three conditions are true.
 - the current picture and the preceding reference picture belong to consecutive access units in decoding order
 - the current picture and the preceding reference picture are reference fields having opposite parity
 - one or more of the following conditions is true
 - the preceding reference picture is an IDR picture
 - the preceding reference picture includes a **memory_management_control_operation** syntax element equal to 5
NOTE 1 – When the preceding reference picture includes a **memory_management_control_operation** syntax element equal to 5, **PrevRefFrameNum** is equal to 0.
 - there is a primary coded picture that precedes the preceding reference picture and the primary coded picture that precedes the preceding reference picture does not have **frame_num** equal to **PrevRefFrameNum**
 - there is a primary coded picture that precedes the preceding reference picture and the primary coded picture that precedes the preceding reference picture is not a reference picture

When the value of **frame_num** is not equal to **PrevRefFrameNum**, the following applies.

- There shall not be any previous field or frame in decoding order that is currently marked as "used for short-term reference" that has a value of **frame_num** equal to any value taken on by the variable **UnusedShortTermFrameNum** in the following:

```
UnusedShortTermFrameNum = ( PrevRefFrameNum + 1 ) % MaxFrameNum
while( UnusedShortTermFrameNum != frame_num )
    UnusedShortTermFrameNum = ( UnusedShortTermFrameNum + 1 ) % MaxFrameNum
```

(7-21)

- The value of **frame_num** is constrained as follows.
 - If **gaps_in_frame_num_value_allowed_flag** is equal to 0, the value of **frame_num** for the current picture shall be equal to $(\text{PrevRefFrameNum} + 1) \% \text{MaxFrameNum}$.
 - Otherwise (**gaps_in_frame_num_value_allowed_flag** is equal to 1), the following applies.
 - If **frame_num** is greater than **PrevRefFrameNum**, there shall not be any non-reference pictures in the bitstream that follow the previous reference picture and precede the current picture in decoding order in which either of the following conditions is true.
 - The value of **frame_num** for the non-reference picture is less than **PrevRefFrameNum**.
 - The value of **frame_num** for the non-reference picture is greater than the value of **frame_num** for the current picture.

- Otherwise (frame_num is less than PrevRefFrameNum), there shall not be any non-reference pictures in the bitstream that follow the previous reference picture and precede the current picture in decoding order in which both of the following conditions are true.
 - The value of frame_num for the non-reference picture is less than PrevRefFrameNum.
 - The value of frame_num for the non-reference picture is greater than the value of frame_num for the current picture.

A picture including a memory_management_control_operation equal to 5 shall have frame_num constraints as described above and, after the decoding of the current picture and the processing of the memory management control operations, the picture shall be inferred to have had frame_num equal to 0 for all subsequent use in the decoding process, except as specified in subclause 7.4.1.2.4.

NOTE 2 – When the primary coded picture is not an IDR picture and does not contain memory_management_control_operation syntax element equal to 5, the value of frame_num of a corresponding redundant coded picture is the same as the value of frame_num in the primary coded picture. Alternatively, the redundant coded picture includes a memory_management_control_operation syntax element equal to 5 and the corresponding primary coded picture is an IDR picture.

field_pic_flag equal to 1 specifies that the slice is a slice of a coded field. field_pic_flag equal to 0 specifies that the slice is a slice of a coded frame. When field_pic_flag is not present it shall be inferred to be equal to 0.

The variable MbaffFrameFlag is derived as follows.

$$\text{MbaffFrameFlag} = (\text{mb_adaptive_frame_field_flag} \ \&\& \ !\text{field_pic_flag}) \quad (7-22)$$

The variable for the picture height in units of macroblocks is derived as follows

$$\text{PicHeightInMbs} = \text{FrameHeightInMbs} / (1 + \text{field_pic_flag}) \quad (7-23)$$

The variable for picture height for the luma component is derived as follows

$$\text{PicHeightInSamples}_L = \text{PicHeightInMbs} * 16 \quad (7-24)$$

The variable for picture height for the chroma component is derived as follows

$$\text{PicHeightInSamples}_C = \text{PicHeightInMbs} * \text{MbHeightC} \quad (7-25)$$

The variable PicSizeInMbs for the current picture is derived according to:

$$\text{PicSizeInMbs} = \text{PicWidthInMbs} * \text{PicHeightInMbs} \quad (7-26)$$

The variable MaxPicNum is derived as follows.

- If field_pic_flag is equal to 0, MaxPicNum is set equal to MaxFrameNum.
- Otherwise (field_pic_flag is equal to 1), MaxPicNum is set equal to 2*MaxFrameNum.

The variable CurrPicNum is derived as follows.

- If field_pic_flag is equal to 0, CurrPicNum is set equal to frame_num.
- Otherwise (field_pic_flag is equal to 1), CurrPicNum is set equal to 2 * frame_num + 1.

bottom_field_flag equal to 1 specifies that the slice is part of a coded bottom field. bottom_field_flag equal to 0 specifies that the picture is a coded top field. When this syntax element is not present for the current slice, it shall be inferred to be equal to 0.

idr_pic_id identifies an IDR picture. The values of idr_pic_id in all the slices of an IDR picture shall remain unchanged. When two consecutive access units in decoding order are both IDR access units, the value of idr_pic_id in the slices of the first such IDR access unit shall differ from the idr_pic_id in the second such IDR access unit. The value of idr_pic_id shall be in the range of 0 to 65535, inclusive.

pic_order_cnt_lsb specifies the picture order count modulo MaxPicOrderCntLsb for the top field of a coded frame or for a coded field. The size of the pic_order_cnt_lsb syntax element is log2_max_pic_order_cnt_lsb_minus4 + 4 bits. The value of the pic_order_cnt_lsb shall be in the range of 0 to MaxPicOrderCntLsb – 1, inclusive.

delta_pic_order_cnt_bottom specifies the picture order count difference between the bottom field and the top field of a coded frame as follows.

- If the current picture includes a `memory_management_control_operation` equal to 5, the value of `delta_pic_order_cnt_bottom` shall be in the range of $(1 - \text{MaxPicOrderCntLsb})$ to $2^{31} - 1$, inclusive.
- Otherwise (the current picture does not include a `memory_management_control_operation` equal to 5), the value of `delta_pic_order_cnt_bottom` shall be in the range of -2^{31} to $2^{31} - 1$, inclusive.

When this syntax element is not present in the bitstream for the current slice, it shall be inferred to be equal to 0.

delta_pic_order_cnt[0] specifies the picture order count difference from the expected picture order count for the top field of a coded frame or for a coded field as specified in subclause 8.2.1. The value of `delta_pic_order_cnt[0]` shall be in the range of -2^{31} to $2^{31} - 1$, inclusive. When this syntax element is not present in the bitstream for the current slice, it shall be inferred to be equal to 0.

delta_pic_order_cnt[1] specifies the picture order count difference from the expected picture order count for the bottom field of a coded frame specified in subclause 8.2.1. The value of `delta_pic_order_cnt[1]` shall be in the range of -2^{31} to $2^{31} - 1$, inclusive. When this syntax element is not present in the bitstream for the current slice, it shall be inferred to be equal to 0.

redundant_pic_cnt shall be equal to 0 for slices and slice data partitions belonging to the primary coded picture. The value of `redundant_pic_cnt` shall be greater than 0 for coded slices or coded slice data partitions of a redundant coded picture. When `redundant_pic_cnt` is not present in the bitstream, its value shall be inferred to be equal to 0. The value of `redundant_pic_cnt` shall be in the range of 0 to 127, inclusive.

NOTE 3 – Any area of the decoded primary picture and the corresponding area that would result from application of the decoding process specified in clause 8 for any redundant picture in the same access unit should be visually similar in appearance.

The value of `pic_parameter_set_id` in a coded slice or coded slice data partition of a redundant coded picture shall be such that the value of `pic_order_present_flag` in the picture parameter set in use in a redundant coded picture is equal to the value of `pic_order_present_flag` in the picture parameter set in use in the corresponding primary coded picture.

When present in the primary coded picture and any redundant coded picture, the following syntax elements shall have the same value: `field_pic_flag`, `bottom_field_flag`, and `idr_pic_id`.

When the value of `nal_ref_idc` in one VCL NAL unit of an access unit is equal to 0, the value of `nal_ref_idc` in all other VCL NAL units of the same access unit shall be equal to 0.

NOTE 4 – The above constraint also has the following implications. If the value of `nal_ref_idc` for the VCL NAL units of the primary coded picture is equal to 0, the value of `nal_ref_idc` for the VCL NAL units of any corresponding redundant coded picture are equal to 0; otherwise (the value of `nal_ref_idc` for the VCL NAL units of the primary coded picture is greater than 0), the value of `nal_ref_idc` for the VCL NAL units of any corresponding redundant coded picture are also greater than 0.

The marking status of reference pictures and the value of `frame_num` after the decoded reference picture marking process as specified in subclause 8.2.5 is invoked for the primary coded picture or any redundant coded picture of the same access unit shall be identical regardless whether the primary coded picture or any redundant coded picture (instead of the primary coded picture) of the access unit would be decoded.

NOTE 5 – The above constraint also has the following implications.

If a primary coded picture is not an IDR picture, the contents of the `dec_ref_pic_marking()` syntax structure must be identical in all slice headers of the primary coded picture and all redundant coded pictures corresponding to the primary coded picture.

Otherwise (a primary coded picture is an IDR picture), the following applies.

If a redundant coded picture corresponding to the primary coded picture is an IDR picture, the contents of the `dec_ref_pic_marking()` syntax structure must be identical in all slice headers of the primary coded picture and the redundant coded picture corresponding to the primary coded picture.

Otherwise (a redundant picture corresponding to the primary coded picture is not an IDR picture), all slice headers of the redundant picture must contain a `dec_ref_pic_marking` syntax() structure including a `memory_management_control_operation` syntax element equal to 5, and the following applies.

If the value of `long_term_reference_flag` in the primary coded picture is equal to 0, the `dec_ref_pic_marking` syntax structure of the redundant coded picture must not include a `memory_management_control_operation` syntax element equal to 6.

Otherwise (the value of `long_term_reference_flag` in the primary coded picture is equal to 1), the `dec_ref_pic_marking` syntax structure of the redundant coded picture must include `memory_management_control_operation` syntax elements equal to 5, 4, and 6 in decoding order, and the value of `max_long_term_frame_idx_plus1` must be equal to 1, and the value of `long_term_frame_idx` must be equal to 0.

The values of `TopFieldOrderCnt` and `BottomFieldOrderCnt` (if applicable) that result after completion of the decoding process for any redundant coded picture or the primary coded picture of the same access unit shall be identical regardless whether the primary coded picture or any redundant coded picture (instead of the primary coded picture) of the access unit would be decoded.

There is no required decoding process for a coded slice or coded slice data partition of a redundant coded picture. When the `redundant_pic_cnt` in the slice header of a coded slice is greater than 0, the decoder may discard the coded slice. However, a coded slice or coded slice data partition of any redundant coded picture shall obey the same constraints as a coded slice or coded slice data partition of a primary picture.

NOTE 6 – When some of the samples in the decoded primary picture cannot be correctly decoded due to errors or losses in transmission of the sequence and a coded redundant slice can be correctly decoded, the decoder should replace the samples of the decoded primary picture with the corresponding samples of the decoded redundant slice. When more than one redundant slice covers the relevant region of the primary picture, the redundant slice having the lowest value of `redundant_pic_cnt` should be used.

Redundant slices and slice data partitions having the same value of `redundant_pic_cnt` belong to the same redundant picture. Decoded slices within the same redundant picture need not cover the entire picture area and shall not overlap.

direct_spatial_mv_pred_flag specifies the method used in the decoding process to derive motion vectors and reference indices for inter prediction as follows.

- If `direct_spatial_mv_pred_flag` is equal to 1, the derivation process for luma motion vectors for `B_Skip`, `B_Direct_16x16`, and `B_Direct_8x8` in subclause 8.4.1.2 shall use spatial direct mode prediction as specified in subclause 8.4.1.2.2.
- Otherwise (`direct_spatial_mv_pred_flag` is equal to 0), the derivation process for luma motion vectors for `B_Skip`, `B_Direct_16x16`, and `B_Direct_8x8` in subclause 8.4.1.2 shall use temporal direct mode prediction as specified in subclause 8.4.1.2.3.

num_ref_idx_active_override_flag equal to 0 specifies that the values of the syntax elements `num_ref_idx_l0_active_minus1` and `num_ref_idx_l1_active_minus1` specified in the referred picture parameter set are in effect. `num_ref_idx_active_override_flag` equal to 1 specifies that the `num_ref_idx_l0_active_minus1` and `num_ref_idx_l1_active_minus1` specified in the referred picture parameter set are overridden for the current slice (and only for the current slice) by the following values in the slice header.

When the current slice is a P, SP, or B slice and `field_pic_flag` is equal to 0 and the value of `num_ref_idx_l0_active_minus1` in the picture parameter set exceeds 15, `num_ref_idx_active_override_flag` shall be equal to 1.

When the current slice is a B slice and `field_pic_flag` is equal to 0 and the value of `num_ref_idx_l1_active_minus1` in the picture parameter set exceeds 15, `num_ref_idx_active_override_flag` shall be equal to 1.

num_ref_idx_l0_active_minus1 specifies the maximum reference index for reference picture list 0 that shall be used to decode the slice.

The range of `num_ref_idx_l0_active_minus1` is specified as follows.

- If `field_pic_flag` is equal to 0, `num_ref_idx_l0_active_minus1` shall be in the range of 0 to 15, inclusive. When `MbaffFrameFlag` is equal to 1, `num_ref_idx_l0_active_minus1` is the maximum index value for the decoding of frame macroblocks and $2 * \text{num_ref_idx_l0_active_minus1} + 1$ is the maximum index value for the decoding of field macroblocks.
- Otherwise (`field_pic_flag` is equal to 1), `num_ref_idx_l0_active_minus1` shall be in the range of 0 to 31, inclusive.

num_ref_idx_l1_active_minus1 has the same semantics as `num_ref_idx_l0_active_minus1` with l0 and list 0 replaced by l1 and list 1, respectively.

cabac_init_idc specifies the index for determining the initialisation table used in the initialisation process for context variables. The value of `cabac_init_idc` shall be in the range of 0 to 2, inclusive.

slice_qp_delta specifies the initial value of QP_Y to be used for all the macroblocks in the slice until modified by the value of `mb_qp_delta` in the macroblock layer. The initial QP_Y quantisation parameter for the slice is computed as:

$$\text{SliceQP}_Y = 26 + \text{pic_init_qp_minus26} + \text{slice_qp_delta} \quad (7-27)$$

The value of `slice_qp_delta` shall be limited such that SliceQP_Y is in the range of $-\text{QpBdOffset}_Y$ to +51, inclusive.

sp_for_switch_flag specifies the decoding process to be used to decode P macroblocks in an SP slice as follows.

- If `sp_for_switch_flag` is equal to 0, the P macroblocks in the SP slice shall be decoded using the SP decoding process for non-switching pictures as specified in subclause 8.6.1.
- Otherwise (`sp_for_switch_flag` is equal to 1), the P macroblocks in the SP slice shall be decoded using the SP and SI decoding process for switching pictures as specified in subclause 8.6.2.

slice_qs_delta specifies the value of QS_Y for all the macroblocks in SP and SI slices. The QS_Y quantisation parameter for the slice is computed as:

$$QS_Y = 26 + \text{pic_init_qs_minus26} + \text{slice_qs_delta} \quad (7-28)$$

The value of `slice_qs_delta` shall be limited such that QS_Y is in the range of 0 to 51, inclusive. This value of QS_Y is used for the decoding of all macroblocks in SI slices with `mb_type` equal to SI and all macroblocks in SP slices with prediction mode equal to inter.

disable_deblocking_filter_idc specifies whether the operation of the deblocking filter shall be disabled across some block edges of the slice and specifies for which edges the filtering is disabled. When `disable_deblocking_filter_idc` is not present in the slice header, the value of `disable_deblocking_filter_idc` shall be inferred to be equal to 0.

The value of `disable_deblocking_filter_idc` shall be in the range of 0 to 2, inclusive.

slice_alpha_c0_offset_div2 specifies the offset used in accessing the α and t_{c0} deblocking filter tables for filtering operations controlled by the macroblocks within the slice. From this value, the offset that shall be applied when addressing these tables shall be computed as:

$$\text{FilterOffsetA} = \text{slice_alpha_c0_offset_div2} \ll 1 \quad (7-29)$$

The value of `slice_alpha_c0_offset_div2` shall be in the range of -6 to +6, inclusive. When `slice_alpha_c0_offset_div2` is not present in the slice header, the value of `slice_alpha_c0_offset_div2` shall be inferred to be equal to 0.

slice_beta_offset_div2 specifies the offset used in accessing the β deblocking filter table for filtering operations controlled by the macroblocks within the slice. From this value, the offset that is applied when addressing the β table of the deblocking filter shall be computed as:

$$\text{FilterOffsetB} = \text{slice_beta_offset_div2} \ll 1 \quad (7-30)$$

The value of `slice_beta_offset_div2` shall be in the range of -6 to +6, inclusive. When `slice_beta_offset_div2` is not present in the slice header the value of `slice_beta_offset_div2` shall be inferred to be equal to 0.

slice_group_change_cycle is used to derive the number of slice group map units in slice group 0 when `slice_group_map_type` is equal to 3, 4, or 5, as specified by

$$\text{MapUnitsInSliceGroup0} = \text{Min}(\text{slice_group_change_cycle} * \text{SliceGroupChangeRate}, \text{PicSizeInMapUnits}) \quad (7-31)$$

The value of `slice_group_change_cycle` is represented in the bitstream by the following number of bits

$$\text{Ceil}(\text{Log2}(\text{PicSizeInMapUnits} \div \text{SliceGroupChangeRate} + 1)) \quad (7-32)$$

The value of `slice_group_change_cycle` shall be in the range of 0 to $\text{Ceil}(\text{PicSizeInMapUnits} \div \text{SliceGroupChangeRate})$, inclusive.

7.4.3.1 Reference picture list reordering semantics

The syntax elements `reordering_of_pic_nums_idc`, `abs_diff_pic_num_minus1`, and `long_term_pic_num` specify the change from the initial reference picture lists to the reference picture lists to be used for decoding the slice.

ref_pic_list_reordering_flag_l0 equal to 1 specifies that the syntax element `reordering_of_pic_nums_idc` is present for specifying reference picture list 0. `ref_pic_list_reordering_flag_l0` equal to 0 specifies that this syntax element is not present.

When `ref_pic_list_reordering_flag_l0` is equal to 1, the number of times that `reordering_of_pic_nums_idc` is not equal to 3 following `ref_pic_list_reordering_flag_l0` shall not exceed `num_ref_idx_l0_active_minus1 + 1`.

When `RefPicList0[num_ref_idx_l0_active_minus1]` in the initial reference picture list produced as specified in subclause 8.2.4.2 is equal to "no reference picture", `ref_pic_list_reordering_flag_l0` shall be equal to 1 and `reordering_of_pic_nums_idc` shall not be equal to 3 until `RefPicList0[num_ref_idx_l0_active_minus1]` in the reordered list produced as specified in subclause 8.2.4.3 is not equal to "no reference picture".

ref_pic_list_reordering_flag_l1 equal to 1 specifies that the syntax element **reordering_of_pic_nums_idc** is present for specifying reference picture list 1. **ref_pic_list_reordering_flag_l1** equal to 0 specifies that this syntax element is not present.

When **ref_pic_list_reordering_flag_l1** is equal to 1, the number of times that **reordering_of_pic_nums_idc** is not equal to 3 following **ref_pic_list_reordering_flag_l1** shall not exceed **num_ref_idx_l1_active_minus1** + 1.

When decoding a B slice and **RefPicList1[num_ref_idx_l1_active_minus1]** in the initial reference picture list produced as specified in subclause 8.2.4.2 is equal to "no reference picture", **ref_pic_list_reordering_flag_l1** shall be equal to 1 and **reordering_of_pic_nums_idc** shall not be equal to 3 until **RefPicList1[num_ref_idx_l1_active_minus1]** in the reordered list produced as specified in subclause 8.2.4.3 is not equal to "no reference picture".

reordering_of_pic_nums_idc together with **abs_diff_pic_num_minus1** or **long_term_pic_num** specifies which of the reference pictures are re-mapped. The values of **reordering_of_pic_nums_idc** are specified in Table 7-7. The value of the first **reordering_of_pic_nums_idc** that follows immediately after **ref_pic_list_reordering_flag_l0** or **ref_pic_list_reordering_flag_l1** shall not be equal to 3.

Table 7-7 – reordering_of_pic_nums_idc operations for reordering of reference picture lists

reordering_of_pic_nums_idc	Reordering specified
0	abs_diff_pic_num_minus1 is present and corresponds to a difference to subtract from a picture number prediction value
1	abs_diff_pic_num_minus1 is present and corresponds to a difference to add to a picture number prediction value
2	long_term_pic_num is present and specifies the long-term picture number for a reference picture
3	End loop for reordering of the initial reference picture list

abs_diff_pic_num_minus1 plus 1 specifies the absolute difference between the picture number of the picture being moved to the current index in the list and the picture number prediction value. **abs_diff_pic_num_minus1** shall be in the range of 0 to **MaxPicNum** – 1. The allowed values of **abs_diff_pic_num_minus1** are further restricted as specified in subclause 8.2.4.3.1.

long_term_pic_num specifies the long-term picture number of the picture being moved to the current index in the list. When decoding a coded frame, **long_term_pic_num** shall be equal to a **LongTermPicNum** assigned to one of the reference frames or complementary reference field pairs marked as "used for long-term reference". When decoding a coded field, **long_term_pic_num** shall be equal to a **LongTermPicNum** assigned to one of the reference fields marked as "used for long-term reference".

7.4.3.2 Prediction weight table semantics

luma_log2_weight_denom is the base 2 logarithm of the denominator for all luma weighting factors. The value of **luma_log2_weight_denom** shall be in the range of 0 to 7, inclusive.

chroma_log2_weight_denom is the base 2 logarithm of the denominator for all chroma weighting factors. The value of **chroma_log2_weight_denom** shall be in the range of 0 to 7, inclusive.

luma_weight_l0_flag equal to 1 specifies that weighting factors for the luma component of list 0 prediction are present. **luma_weight_l0_flag** equal to 0 specifies that these weighting factors are not present.

luma_weight_l0[i] is the weighting factor applied to the luma prediction value for list 0 prediction using **RefPicList0[i]**. When **luma_weight_l0_flag** is equal to 1, the value of **luma_weight_l0[i]** shall be in the range of –128 to 127, inclusive. When **luma_weight_l0_flag** is equal to 0, **luma_weight_l0[i]** shall be inferred to be equal to $2^{\text{luma_log2_weight_denom}}$ for **RefPicList0[i]**.

luma_offset_l0[i] is the additive offset applied to the luma prediction value for list 0 prediction using **RefPicList0[i]**. The value of **luma_offset_l0[i]** shall be in the range of –128 to 127, inclusive. When **luma_weight_l0_flag** is equal to 0, **luma_offset_l0[i]** shall be inferred as equal to 0 for **RefPicList0[i]**.

chroma_weight_l0_flag equal to 1 specifies that weighting factors for the chroma prediction values of list 0 prediction are present. **chroma_weight_l0_flag** equal to 0 specifies that these weighting factors are not present.

chroma_weight_l0[i][j] is the weighting factor applied to the chroma prediction values for list 0 prediction using RefPicList0[i] with j equal to 0 for Cb and j equal to 1 for Cr. When chroma_weight_l0_flag is equal to 1, the value of chroma_weight_l0[i][j] shall be in the range of -128 to 127, inclusive. When chroma_weight_l0_flag is equal to 0, chroma_weight_l0[i][j] shall be inferred to be equal to $2^{\text{chroma_log2_weight_denom}}$ for RefPicList0[i].

chroma_offset_l0[i][j] is the additive offset applied to the chroma prediction values for list 0 prediction using RefPicList0[i] with j equal to 0 for Cb and j equal to 1 for Cr. The value of chroma_offset_l0[i][j] shall be in the range of -128 to 127, inclusive. When chroma_weight_l0_flag is equal to 0, chroma_offset_l0[i][j] shall be inferred to be equal to 0 for RefPicList0[i].

luma_weight_l1_flag, luma_weight_l1, luma_offset_l1, chroma_weight_l1_flag, chroma_weight_l1, chroma_offset_l1 have the same semantics as luma_weight_l0_flag, luma_weight_l0, luma_offset_l0, chroma_weight_l0_flag, chroma_weight_l0, chroma_offset_l0, respectively, with l0, list 0, and List0 replaced by l1, list 1, and List1, respectively.

7.4.3.3 Decoded reference picture marking semantics

The syntax elements no_output_of_prior_pics_flag, long_term_reference_flag, adaptive_ref_pic_marking_mode_flag, memory_management_control_operation, difference_of_pic_nums_minus1, long_term_frame_idx, long_term_pic_num, and max_long_term_frame_idx_plus1 specify marking of the reference pictures.

The marking of a reference picture can be "unused for reference", "used for short-term reference", or "used for long-term reference", but only one among these three. When a reference picture is referred to as being marked as "used for reference", this collectively refers to the picture being marked as "used for short-term reference" or "used for long-term reference" (but not both). A reference picture that is marked as "used for short-term reference" is referred to as a short-term reference picture. A reference picture that is marked as "used for long-term reference" is referred to as a long-term reference picture.

The syntax element adaptive_ref_pic_marking_mode_flag and the content of the decoded reference picture marking syntax structure shall be identical for all coded slices of a coded picture.

The syntax category of the decoded reference picture marking syntax structure shall be inferred as follows.

- If the decoded reference picture marking syntax structure is in a slice header, the syntax category of the decoded reference picture marking syntax structure shall be inferred to be equal to 2.
- Otherwise (the decoded reference picture marking syntax structure is in a decoded reference picture marking repetition SEI message as specified in Annex D), the syntax category of the decoded reference picture marking syntax structure shall be inferred to be equal to 5.

no_output_of_prior_pics_flag specifies how the previously-decoded pictures in the decoded picture buffer are treated after decoding of an IDR picture. See Annex C. When the IDR picture is the first IDR picture in the bitstream, the value of no_output_of_prior_pics_flag has no effect on the decoding process. When the IDR picture is not the first IDR picture in the bitstream and the value of PicWidthInMbs, FrameHeightInMbs, or max_dec_frame_buffering derived from the active sequence parameter set is different from the value of PicWidthInMbs, FrameHeightInMbs, or max_dec_frame_buffering derived from the sequence parameter set active for the preceding sequence, no_output_of_prior_pics_flag equal to 1 may be inferred by the decoder, regardless of the actual value of no_output_of_prior_pics_flag.

long_term_reference_flag equal to 0 specifies that the MaxLongTermFrameIdx variable is set equal to "no long-term frame indices" and that the IDR picture is marked as "used for short-term reference". long_term_reference_flag equal to 1 specifies that the MaxLongTermFrameIdx variable is set equal to 0 and that the current IDR picture is marked "used for long-term reference" and is assigned LongTermFrameIdx equal to 0. When num_ref_frames is equal to 0, long_term_reference_flag shall be equal to 0.

adaptive_ref_pic_marking_mode_flag selects the reference picture marking mode of the currently decoded picture as specified in Table 7-8. adaptive_ref_pic_marking_mode_flag shall be equal to 1 when the number of frames, complementary field pairs, and non-paired fields that are currently marked as "used for long-term reference" is equal to Max(num_ref_frames, 1).

Table 7-8 – Interpretation of adaptive_ref_pic_marking_mode_flag

adaptive_ref_pic_marking_mode_flag	Reference picture marking mode specified
0	Sliding window reference picture marking mode: A marking mode providing a first-in first-out mechanism for short-term reference pictures.
1	Adaptive reference picture marking mode: A reference picture marking mode providing syntax elements to specify marking of reference pictures as “unused for reference” and to assign long-term frame indices.

memory_management_control_operation specifies a control operation to be applied to affect the reference picture marking. The `memory_management_control_operation` syntax element is followed by data necessary for the operation specified by the value of `memory_management_control_operation`. The values and control operations associated with `memory_management_control_operation` are specified in Table 7-9. The `memory_management_control_operation` syntax elements are processed by the decoding process in the order in which they appear in the slice header, and the semantics constraints expressed for each `memory_management_control_operation` apply at the specific position in that order at which that individual `memory_management_control_operation` is processed.

For interpretation of `memory_management_control_operation`, the term reference picture is interpreted as follows.

- If the current picture is a frame, the term reference picture refers either to a reference frame or a complementary reference field pair.
- Otherwise (the current picture is a field), the term reference picture refers either to a reference field or a field of a reference frame.

`memory_management_control_operation` shall not be equal to 1 in a slice header unless the specified reference picture is marked as "used for short-term reference" when the `memory_management_control_operation` is processed by the decoding process.

`memory_management_control_operation` shall not be equal to 2 in a slice header unless the specified long-term picture number refers to a reference picture that is marked as "used for long-term reference" when the `memory_management_control_operation` is processed by the decoding process.

`memory_management_control_operation` shall not be equal to 3 in a slice header unless the specified reference picture is marked as "used for short-term reference" when the `memory_management_control_operation` is processed by the decoding process.

`memory_management_control_operation` shall not be equal to 3 or 6 if the value of the variable `MaxLongTermFrameIdx` is equal to "no long-term frame indices" when the `memory_management_control_operation` is processed by the decoding process.

Not more than one `memory_management_control_operation` equal to 4 shall be present in a slice header.

Not more than one `memory_management_control_operation` equal to 5 shall be present in a slice header.

Not more than one `memory_management_control_operation` equal to 6 shall be present in a slice header.

`memory_management_control_operation` shall not be equal to 5 in a slice header unless no `memory_management_control_operation` in the range of 1 to 3 is present in the same decoded reference picture marking syntax structure.

A `memory_management_control_operation` equal to 5 shall not follow a `memory_management_control_operation` equal to 6 in the same slice header.

When a `memory_management_control_operation` equal to 6 is present, any `memory_management_control_operation` equal to 2, 3, or 4 that follows the `memory_management_control_operation` equal to 6 within the same slice header shall not specify the current picture to be marked as "unused for reference".

NOTE 1 – These constraints prohibit any combination of multiple `memory_management_control_operation` syntax elements that would specify the current picture to be marked as "unused for reference". However, some other combinations of `memory_management_control_operation` syntax elements are permitted that may affect the marking status of other reference pictures more than once in the same slice header. In particular, it is permitted for a `memory_management_control_operation` equal to 3 that specifies a long-term frame index to be assigned to a particular short-term reference picture to be followed in the same slice header by a `memory_management_control_operation` equal to 2, 3, 4 or 6 that specifies the same reference picture to subsequently be marked as "unused for reference".

Table 7-9 – Memory management control operation (memory_management_control_operation) values

memory_management_control_operation	Memory Management Control Operation
0	End memory_management_control_operation syntax element loop
1	Mark a short-term reference picture as "unused for reference"
2	Mark a long-term reference picture as "unused for reference"
3	Mark a short-term reference picture as "used for long-term reference" and assign a long-term frame index to it
4	Specify the maximum long-term frame index and mark all long-term reference pictures having long-term frame indices greater than the maximum value as "unused for reference"
5	Mark all reference pictures as "unused for reference" and set the MaxLongTermFrameIdx variable to "no long-term frame indices"
6	Mark the current picture as "used for long-term reference" and assign a long-term frame index to it

When decoding a field and a memory_management_control_operation command equal to 3 is present that assigns a long-term frame index to a field that is part of a short-term reference frame or part of a short-term complementary reference field pair, another memory_management_control_operation command to assign the same long-term frame index to the other field of the same frame or complementary reference field pair shall be present in the same decoded reference picture marking syntax structure.

NOTE 2 – The above requirement must be fulfilled even when the field referred to by the memory_management_control_operation equal to 3 is subsequently marked as "unused for reference" (for example when a memory_management_control_operation equal to 2 is present in the same slice header that causes the field to be marked as "unused for reference").

When the first field (in decoding order) of a complementary reference field pair includes a long_term_reference_flag equal to 1 or a memory_management_control_operation command equal to 6, the decoded reference picture marking syntax structure for the other field of the complementary reference field pair shall contain a memory_management_control_operation command equal to 6 that assigns the same long-term frame index to the other field.

NOTE 3 – The above requirement must be fulfilled even when the first field of the complementary reference field pair is subsequently marked as "unused for reference" (for example, when a memory_management_control_operation equal to 2 is present in the slice header of the second field that causes the first field to be marked as "unused for reference").

difference_of_pic_nums_minus1 is used (with memory_management_control_operation equal to 3 or 1) to assign a long-term frame index to a short-term reference picture or to mark a short-term reference picture as "unused for reference". When the associated memory_management_control_operation is processed by the decoding process, the resulting picture number derived from difference_of_pic_nums_minus1 shall be a picture number assigned to one of the reference pictures marked as "used for reference" and not previously assigned to a long-term frame index.

The resulting picture number is constrained as follows.

- If field_pic_flag is equal to 0, the resulting picture number shall be one of the set of picture numbers assigned to reference frames or complementary reference field pairs.

NOTE 4 – When field_pic_flag is equal to 0, the resulting picture number must be a picture number assigned to a complementary reference field pair in which both fields are marked as "used for reference" or a frame in which both fields are marked as "used for reference". In particular, when field_pic_flag is equal to 0, the marking of a non-paired field or a frame in which a single field is marked as "used for reference" cannot be affected by a memory_management_control_operation equal to 1.

- Otherwise (field_pic_flag is equal to 1), the resulting picture number shall be one of the set of picture numbers assigned to reference fields.

long_term_pic_num is used (with `memory_management_control_operation` equal to 2) to mark a long-term reference picture as "unused for reference". When the associated `memory_management_control_operation` is processed by the decoding process, `long_term_pic_num` shall be equal to a long-term picture number assigned to one of the reference pictures that is currently marked as "used for long-term reference".

The resulting long-term picture number is constrained as follows.

- If `field_pic_flag` is equal to 0, the resulting long-term picture number shall be one of the set of long-term picture numbers assigned to reference frames or complementary reference field pairs.

NOTE 5 – When `field_pic_flag` is equal to 0, the resulting long-term picture number must be a long-term picture number assigned to a complementary reference field pair in which both fields are marked as "used for reference" or a frame in which both fields are marked as "used for reference". In particular, when `field_pic_flag` is equal to 0, the marking of a non-paired field or a frame in which a single field is marked as "used for reference" cannot be affected by a `memory_management_control_operation` equal to 2.

- Otherwise (`field_pic_flag` is equal to 1), the resulting long-term picture number shall be one of the set of long-term picture numbers assigned to reference fields.

long_term_frame_idx is used (with `memory_management_control_operation` equal to 3 or 6) to assign a long-term frame index to a picture. When the associated `memory_management_control_operation` is processed by the decoding process, the value of `long_term_frame_idx` shall be in the range of 0 to `MaxLongTermFrameIdx`, inclusive.

max_long_term_frame_idx_plus1 minus 1 specifies the maximum value of long-term frame index allowed for long-term reference pictures (until receipt of another value of `max_long_term_frame_idx_plus1`). The value of `max_long_term_frame_idx_plus1` shall be in the range of 0 to `num_ref_frames`, inclusive.

7.4.4 Slice data semantics

cabac_alignment_one_bit is a bit equal to 1.

mb_skip_run specifies the number of consecutive skipped macroblocks for which, when decoding a P or SP slice, `mb_type` shall be inferred to be `P_Skip` and the macroblock type is collectively referred to as a P macroblock type, or for which, when decoding a B slice, `mb_type` shall be inferred to be `B_Skip` and the macroblock type is collectively referred to as a B macroblock type. The value of `mb_skip_run` shall be in the range of 0 to `PicSizeInMbs – CurrMbAddr`, inclusive.

mb_skip_flag equal to 1 specifies that for the current macroblock, when decoding a P or SP slice, `mb_type` shall be inferred to be `P_Skip` and the macroblock type is collectively referred to as P macroblock type, or for which, when decoding a B slice, `mb_type` shall be inferred to be `B_Skip` and the macroblock type is collectively referred to as B macroblock type. `mb_skip_flag` equal to 0 specifies that the current macroblock is not skipped.

mb_field_decoding_flag equal to 0 specifies that the current macroblock pair is a frame macroblock pair. `mb_field_decoding_flag` equal to 1 specifies that the macroblock pair is a field macroblock pair. Both macroblocks of a frame macroblock pair are referred to in the text as frame macroblocks, whereas both macroblocks of a field macroblock pair are referred to in the text as field macroblocks.

When `mb_field_decoding_flag` is not present for either macroblock of a macroblock pair, the value of `mb_field_decoding_flag` is derived as follows.

- If there is a neighbouring macroblock pair immediately to the left of the current macroblock pair in the same slice, the value of `mb_field_decoding_flag` shall be inferred to be equal to the value of `mb_field_decoding_flag` for the neighbouring macroblock pair immediately to the left of the current macroblock pair,
- Otherwise, if there is no neighbouring macroblock pair immediately to the left of the current macroblock pair in the same slice and there is a neighbouring macroblock pair immediately above the current macroblock pair in the same slice, the value of `mb_field_decoding_flag` shall be inferred to be equal to the value of `mb_field_decoding_flag` for the neighbouring macroblock pair immediately above the current macroblock pair,
- Otherwise (there is no neighbouring macroblock pair either immediately to the left or immediately above the current macroblock pair in the same slice), the value of `mb_field_decoding_flag` shall be inferred to be equal to 0.

end_of_slice_flag equal to 0 specifies that another macroblock is following in the slice. `end_of_slice_flag` equal to 1 specifies the end of the slice and that no further macroblock follows.

The function `NextMbAddress()` used in the slice data syntax table is specified in subclause 8.2.2.

7.4.5 Macroblock layer semantics

mb_type specifies the macroblock type. The semantics of `mb_type` depend on the slice type.

Tables and semantics are specified for the various macroblock types for I, SI, P, SP, and B slices. Each table presents the value of mb_type, the name of mb_type, the number of macroblock partitions used (given by the NumMbPart(mb_type) function), the prediction mode of the macroblock (when it is not partitioned) or the first partition (given by the MbPartPredMode(mb_type, 0) function) and the prediction mode of the second partition (given by the MbPartPredMode(mb_type, 1) function). When a value is not applicable it is designated by “na”. In the text, the value of mb_type may be referred to as the macroblock type and a value X of MbPartPredMode() may be referred to in the text by “X macroblock (partition) prediction mode” or as “X prediction macroblocks”.

Table 7-10 shows the allowed collective macroblock types for each slice_type.

NOTE 1 – There are some macroblock types with Pred_L0 prediction mode that are classified as B macroblock types.

Table 7-10 – Allowed collective macroblock types for slice_type

slice_type	allowed collective macroblock types
I (slice)	I (see Table 7-11) (macroblock types)
P (slice)	P (see Table 7-13) and I (see Table 7-11) (macroblock types)
B (slice)	B (see Table 7-14) and I (see Table 7-11) (macroblock types)
SI (slice)	SI (see Table 7-12) and I (see Table 7-11) (macroblock types)
SP (slice)	P (see Table 7-13) and I (see Table 7-11) (macroblock types)

transform_size_8x8_flag equal to 1 specifies that for the current macroblock the transform coefficient decoding process and picture construction process prior to deblocking filter process for residual 8x8 blocks shall be invoked for luma samples. transform_size_8x8_flag equal to 0 specifies that for the current macroblock the transform coefficient decoding process and picture construction process prior to deblocking filter process for residual 4x4 blocks shall be invoked for luma samples. When transform_size_8x8_flag is not present in the bitstream, it shall be inferred to be equal to 0.

NOTE 2 – When the current macroblock prediction mode MbPartPredMode(mb_type, 0) is equal to Intra_16x16, transform_size_8x8_flag is not present in the bitstream and then inferred to be equal to 0.

When sub_mb_type[mbPartIdx] (see subclause 7.4.5.2) is present in the bitstream for all 8x8 blocks indexed by mbPartIdx = 0..3, the variable noSubMbPartSizeLessThan8x8Flag indicates whether for each of the four 8x8 blocks the corresponding SubMbPartWidth(sub_mb_type[mbPartIdx]) and SubMbPartHeight(sub_mb_type[mbPartIdx]) are both equal to 8.

NOTE 3 – When noSubMbPartSizeLessThan8x8Flag is equal to 0 and the current macroblock type is not equal to I_NxN, transform_size_8x8_flag is not present in the bitstream and then inferred to be equal to 0.

Macroblock types that may be collectively referred to as I macroblock types are specified in Table 7-11.

The macroblock types for I slices are all I macroblock types.

Table 7-11 – Macroblock types for I slices

mb_type	Name of mb_type	transform_size_8x8_flag	MbPartPredMode (mb_type, 0)	Intra16x16PredMode	CodedBlockPatternChroma	CodedBlockPatternLuma
0	I_NxN	0	Intra_4x4	na	Equation 7-33	Equation 7-33
0	I_NxN	1	Intra_8x8	na	Equation 7-33	Equation 7-33
1	I_16x16_0_0_0	na	Intra_16x16	0	0	0
2	I_16x16_1_0_0	na	Intra_16x16	1	0	0
3	I_16x16_2_0_0	na	Intra_16x16	2	0	0
4	I_16x16_3_0_0	na	Intra_16x16	3	0	0
5	I_16x16_0_1_0	na	Intra_16x16	0	1	0
6	I_16x16_1_1_0	na	Intra_16x16	1	1	0
7	I_16x16_2_1_0	na	Intra_16x16	2	1	0
8	I_16x16_3_1_0	na	Intra_16x16	3	1	0
9	I_16x16_0_2_0	na	Intra_16x16	0	2	0
10	I_16x16_1_2_0	na	Intra_16x16	1	2	0
11	I_16x16_2_2_0	na	Intra_16x16	2	2	0
12	I_16x16_3_2_0	na	Intra_16x16	3	2	0
13	I_16x16_0_0_1	na	Intra_16x16	0	0	15
14	I_16x16_1_0_1	na	Intra_16x16	1	0	15
15	I_16x16_2_0_1	na	Intra_16x16	2	0	15
16	I_16x16_3_0_1	na	Intra_16x16	3	0	15
17	I_16x16_0_1_1	na	Intra_16x16	0	1	15
18	I_16x16_1_1_1	na	Intra_16x16	1	1	15
19	I_16x16_2_1_1	na	Intra_16x16	2	1	15
20	I_16x16_3_1_1	na	Intra_16x16	3	1	15
21	I_16x16_0_2_1	na	Intra_16x16	0	2	15
22	I_16x16_1_2_1	na	Intra_16x16	1	2	15
23	I_16x16_2_2_1	na	Intra_16x16	2	2	15
24	I_16x16_3_2_1	na	Intra_16x16	3	2	15
25	I_PCM	na	na	na	na	na

The following semantics are assigned to the macroblock types in Table 7-11.

I_NxN: A mnemonic name for mb_type equal to 0 with MbPartPredMode(mb_type, 0) equal to Intra_4x4 or Intra_8x8.

I_16x16_0_0_0, I_16x16_1_0_0, I_16x16_2_0_0, I_16x16_3_0_0, I_16x16_0_1_0, I_16x16_1_1_0, I_16x16_2_1_0, I_16x16_3_1_0, I_16x16_0_2_0, I_16x16_1_2_0, I_16x16_2_2_0, I_16x16_3_2_0, I_16x16_0_0_1, I_16x16_1_0_1, I_16x16_2_0_1, I_16x16_3_0_1, I_16x16_0_1_1, I_16x16_1_1_1, I_16x16_2_1_1, I_16x16_3_1_1, I_16x16_0_2_1, I_16x16_1_2_1, I_16x16_2_2_1, I_16x16_3_2_1: the macroblock is coded as an Intra_16x16 prediction mode macroblock.

To each Intra_16x16 prediction macroblock, an Intra16x16PredMode is assigned, which specifies the Intra_16x16 prediction mode. CodedBlockPatternChroma contains the coded block pattern value for chroma as specified in Table 7-15. When chroma_format_idc is equal to 0, CodedBlockPatternChroma shall be equal to 0. CodedBlockPatternLuma specifies whether, for the luma component, non-zero AC transform coefficient levels are present. CodedBlockPatternLuma equal to 0 specifies that all AC transform coefficient levels in the luma component of the macroblock are equal to 0. CodedBlockPatternLuma equal to 15 specifies that at least one of the AC transform coefficient levels in the luma component of the macroblock is non-zero, requiring scanning of AC transform coefficient levels for all 16 of the 4x4 blocks in the 16x16 block.

Intra_4x4 specifies the macroblock prediction mode and specifies that the Intra_4x4 prediction process is invoked as specified in subclause 8.3.1. Intra_4x4 is an Intra macroblock prediction mode.

Intra_8x8 specifies the macroblock prediction mode and specifies that the Intra_8x8 prediction process is invoked as specified in subclause 8.3.2. Intra_8x8 is an Intra macroblock prediction mode.

Intra_16x16 specifies the macroblock prediction mode and specifies that the Intra_16x16 prediction process is invoked as specified in subclause 8.3.3. Intra_16x16 is an Intra macroblock prediction mode.

For a macroblock coded with mb_type equal to I_PCM, the Intra macroblock prediction mode shall be inferred.

A macroblock type that may be referred to as SI macroblock type is specified in Table 7-12.

The macroblock types for SI slices are specified in Tables 7-12 and 7-11. The mb_type value 0 is specified in Table 7-12 and the mb_type values 1 to 26 are specified in Table 7-11, indexed by subtracting 1 from the value of mb_type.

Table 7-12 – Macroblock type with value 0 for SI slices

mb_type	Name of mb_type	MbPartPredMode (mb_type, 0)	Intra16x16PredMode	CodedBlockPatternChroma	CodedBlockPatternLuma
0	SI	Intra_4x4	na	Equation 7-33	Equation 7-33

The following semantics are assigned to the macroblock type in Table 7-12. The SI macroblock is coded as Intra_4x4 prediction macroblock.

Macroblock types that may be collectively referred to as P macroblock types are specified in Table 7-13.

The macroblock types for P and SP slices are specified in Tables 7-13 and 7-11. mb_type values 0 to 4 are specified in Table 7-13 and mb_type values 5 to 30 are specified in Table 7-11, indexed by subtracting 5 from the value of mb_type.

Table 7-13 – Macroblock type values 0 to 4 for P and SP slices

mb_type	Name of mb_type	NumMbPart (mb_type)	MbPartPredMode (mb_type, 0)	MbPartPredMode (mb_type, 1)	MbPartWidth (mb_type)	MbPartHeight (mb_type)
0	P_L0_16x16	1	Pred_L0	na	16	16
1	P_L0_L0_16x8	2	Pred_L0	Pred_L0	16	8
2	P_L0_L0_8x16	2	Pred_L0	Pred_L0	8	16
3	P_8x8	4	na	na	8	8
4	P_8x8ref0	4	na	na	8	8
inferred	P_Skip	1	Pred_L0	na	16	16

The following semantics are assigned to the macroblock types in Table 7-13.

- P_L0_16x16: the samples of the macroblock are predicted with one luma macroblock partition of size 16x16 luma samples and associated chroma samples.
- P_L0_L0_MxN, with MxN being replaced by 16x8 or 8x16: the samples of the macroblock are predicted using two luma partitions of size MxN equal to 16x8, or two luma partitions of size MxN equal to 8x16, and associated chroma samples, respectively.
- P_8x8: for each sub-macroblock an additional syntax element (sub_mb_type) is present in the bitstream that specifies the type of the corresponding sub-macroblock (see subclause 7.4.5.2).
- P_8x8ref0: has the same semantics as P_8x8 but no syntax element for the reference index (ref_idx_l0) is present in the bitstream and ref_idx_l0[mbPartIdx] shall be inferred to be equal to 0 for all sub-macroblocks of the macroblock (with indices mbPartIdx equal to 0..3).
- P_Skip: no further data is present for the macroblock in the bitstream.

The following semantics are assigned to the macroblock prediction modes (MbPartPredMode()) in Table 7-13.

- Pred_L0: specifies that the inter prediction process is invoked using list 0 prediction. Pred_L0 is an Inter macroblock prediction mode.

Macroblock types that may be collectively referred to as B macroblock types are specified in Table 7-14.

The macroblock types for B slices are specified in Tables 7-14 and 7-11. The mb_type values 0 to 22 are specified in Table 7-14 and the mb_type values 23 to 48 are specified in Table 7-11, indexed by subtracting 23 from the value of mb_type.

Table 7-14 – Macroblock type values 0 to 22 for B slices

mb_type	Name of mb_type	NumMbPart (mb_type)	MbPartPredMode (mb_type, 0)	MbPartPredMode (mb_type, 1)	MbPartWidth (mb_type)	MbPartHeight (mb_type)
0	B_Direct_16x16	na	Direct	na	8	8
1	B_L0_16x16	1	Pred_L0	na	16	16
2	B_L1_16x16	1	Pred_L1	na	16	16
3	B_Bi_16x16	1	BiPred	na	16	16
4	B_L0_L0_16x8	2	Pred_L0	Pred_L0	16	8
5	B_L0_L0_8x16	2	Pred_L0	Pred_L0	8	16
6	B_L1_L1_16x8	2	Pred_L1	Pred_L1	16	8
7	B_L1_L1_8x16	2	Pred_L1	Pred_L1	8	16
8	B_L0_L1_16x8	2	Pred_L0	Pred_L1	16	8
9	B_L0_L1_8x16	2	Pred_L0	Pred_L1	8	16
10	B_L1_L0_16x8	2	Pred_L1	Pred_L0	16	8
11	B_L1_L0_8x16	2	Pred_L1	Pred_L0	8	16
12	B_L0_Bi_16x8	2	Pred_L0	BiPred	16	8
13	B_L0_Bi_8x16	2	Pred_L0	BiPred	8	16
14	B_L1_Bi_16x8	2	Pred_L1	BiPred	16	8
15	B_L1_Bi_8x16	2	Pred_L1	BiPred	8	16
16	B_Bi_L0_16x8	2	BiPred	Pred_L0	16	8
17	B_Bi_L0_8x16	2	BiPred	Pred_L0	8	16
18	B_Bi_L1_16x8	2	BiPred	Pred_L1	16	8
19	B_Bi_L1_8x16	2	BiPred	Pred_L1	8	16
20	B_Bi_Bi_16x8	2	BiPred	BiPred	16	8
21	B_Bi_Bi_8x16	2	BiPred	BiPred	8	16
22	B_8x8	4	na	na	8	8
inferred	B_Skip	na	Direct	na	8	8

The following semantics are assigned to the macroblock types in Table 7-14:

- B_Direct_16x16: no motion vector differences or reference indices are present for the macroblock in the bitstream. The functions MbPartWidth(B_Direct_16x16), and MbPartHeight(B_Direct_16x16) are used in the derivation process for motion vectors and reference frame indices in subclause 8.4.1 for direct mode prediction.
- B_X_16x16 with X being replaced by L0, L1, or Bi: the samples of the macroblock are predicted with one luma macroblock partition of size 16x16 luma samples and associated chroma samples. For a macroblock with type B_X_16x16 with X being replaced by either L0 or L1, one motion vector difference and one reference index is

present in the bitstream for the macroblock. For a macroblock with type B_X_16x16 with X being replaced by Bi, two motion vector differences and two reference indices are present in the bitstream for the macroblock.

- B_X0_X1_MxN, with X0, X1 referring to the first and second macroblock partition and being replaced by L0, L1, or Bi, and MxN being replaced by 16x8 or 8x16: the samples of the macroblock are predicted using two luma partitions of size MxN equal to 16x8, or two luma partitions of size MxN equal to 8x16, and associated chroma samples, respectively. For a macroblock partition X0 or X1 with X0 or X1 being replaced by either L0 or L1, one motion vector difference and one reference index is present in the bitstream. For a macroblock partition X0 or X1 with X0 or X1 being replaced by Bi, two motion vector differences and two reference indices are present in the bitstream for the macroblock partition.
- B_8x8: for each sub-macroblock an additional syntax element (sub_mb_type) is present in the bitstream that specifies the type of the corresponding sub-macroblock (see subclause 7.4.5.2).
- B_Skip: no further data is present for the macroblock in the bitstream. The functions MbPartWidth(B_Skip), and MbPartHeight(B_Skip) are used in the derivation process for motion vectors and reference frame indices in subclause 8.4.1 for direct mode prediction.

The following semantics are assigned to the macroblock prediction modes (MbPartPredMode()) in Table 7-14.

- Direct: no motion vector differences or reference indices are present for the macroblock (in case of B_Skip or B_Direct_16x16) in the bitstream. Direct is an Inter macroblock prediction mode.
- Pred_L0: see semantics for Table 7-13.
- Pred_L1: specifies that the Inter prediction process is invoked using list 1 prediction. Pred_L1 is an Inter macroblock prediction mode.
- BiPred: specifies that the Inter prediction process is invoked using list 0 and list 1 prediction. BiPred is an Inter macroblock prediction mode.

pcm_alignment_zero_bit is a bit equal to 0.

pcm_sample_luma[i] is a sample value. The first pcm_sample_luma[i] values represent luma sample values in the raster scan within the macroblock. The number of bits used to represent each of these samples is BitDepth_Y. When profile_idc is not equal to 100, 110, 122, or 144, pcm_sample_luma[i] shall not be equal to 0.

pcm_sample_chroma[i] is a sample value. The first MbWidthC * MbHeightC pcm_sample_chroma[i] values represent Cb sample values in the raster scan within the macroblock and the remaining MbWidthC * MbHeightC pcm_sample_chroma[i] values represent Cr sample values in the raster scan within the macroblock. The number of bits used to represent each of these samples is BitDepth_C. When profile_idc is not equal to 100, 110, 122, or 144, pcm_sample_chroma[i] shall not be equal to 0.

coded_block_pattern specifies which of the four 8x8 luma blocks and associated chroma blocks of a macroblock may contain non-zero transform coefficient levels. For macroblocks with prediction mode not equal to Intra_16x16, coded_block_pattern is present in the bitstream and the variables CodedBlockPatternLuma and CodedBlockPatternChroma are derived as follows.

$$\begin{aligned} \text{CodedBlockPatternLuma} &= \text{coded_block_pattern} \% 16 \\ \text{CodedBlockPatternChroma} &= \text{coded_block_pattern} / 16 \end{aligned} \quad (7-33)$$

When coded_block_pattern is present, CodedBlockPatternLuma specifies, for each of the four 8x8 luma blocks of the macroblock, one of the following cases.

- All transform coefficient levels of the four 4x4 luma blocks in the 8x8 luma block are equal to zero
- One or more transform coefficient levels of one or more of the 4x4 luma blocks in the 8x8 luma block shall be non-zero valued.

The meaning of CodedBlockPatternChroma is specified in Table 7-15.

Table 7-15 – Specification of CodedBlockPatternChroma values

CodedBlockPatternChroma	Description
0	All chroma transform coefficient levels are equal to 0.
1	One or more chroma DC transform coefficient levels shall be non-zero valued. All chroma AC transform coefficient levels are equal to 0.
2	Zero or more chroma DC transform coefficient levels are non-zero valued. One or more chroma AC transform coefficient levels shall be non-zero valued.

mb_qp_delta can change the value of QP_Y in the macroblock layer. The decoded value of **mb_qp_delta** shall be in the range of $-(26 + QpBdOffset_Y / 2)$ to $+(25 + QpBdOffset_Y / 2)$, inclusive. **mb_qp_delta** shall be inferred to be equal to 0 when it is not present for any macroblock (including P_Skip and B_Skip macroblock types).

The value of QP_Y is derived as

$$QP_Y = ((QP_{Y,PREV} + mb_qp_delta + 52 + 2 * QpBdOffset_Y) \% (52 + QpBdOffset_Y)) - QpBdOffset_Y \quad (7-34)$$

where $QP_{Y,PREV}$ is the luma quantisation parameter, QP_Y , of the previous macroblock in decoding order in the current slice. For the first macroblock in the slice $QP_{Y,PREV}$ is initially set equal to $SliceQP_Y$ derived in Equation 7-27 at the start of each slice.

The value of QP'_Y is derived as

$$QP'_Y = QP_Y + QpBdOffset_Y \quad (7-35)$$

7.4.5.1 Macroblock prediction semantics

All samples of the macroblock are predicted. The prediction modes are derived using the following syntax elements.

prev_intra4x4_pred_mode_flag[luma4x4BlkIdx] and **rem_intra4x4_pred_mode**[luma4x4BlkIdx] specify the Intra_4x4 prediction of the 4x4 luma block with index luma4x4BlkIdx = 0..15.

prev_intra8x8_pred_mode_flag[luma8x8BlkIdx] and **rem_intra8x8_pred_mode**[luma8x8BlkIdx] specify the Intra_8x8 prediction of the 8x8 luma block with index luma8x8BlkIdx = 0..3.

intra_chroma_pred_mode specifies the type of spatial prediction used for chroma in macroblocks using Intra_4x4 or Intra_16x16 prediction, as shown in Table 7-16. The value of **intra_chroma_pred_mode** shall be in the range of 0 to 3, inclusive.

Table 7-16 – Relationship between intra_chroma_pred_mode and spatial prediction modes

intra_chroma_pred_mode	Intra Chroma Prediction Mode
0	DC
1	Horizontal
2	Vertical
3	Plane

ref_idx_l0[mbPartIdx] when present, specifies the index in reference picture list 0 of the reference picture to be used for prediction.

The range of **ref_idx_l0**[mbPartIdx], the index in list 0 of the reference picture, and, if applicable, the parity of the field within the reference picture used for prediction are specified as follows.

- If MbaffFrameFlag is equal to 0 or mb_field_decoding_flag is equal to 0, the value of **ref_idx_l0**[mbPartIdx] shall be in the range of 0 to num_ref_idx_l0_active_minus1, inclusive.

- Otherwise (MbaffFrameFlag is equal to 1 and mb_field_decoding_flag is equal to 1), the value of ref_idx_l0[mbPartIdx] shall be in the range of 0 to 2 * num_ref_idx_l0_active_minus1 + 1, inclusive.

When only one reference picture is used for inter prediction, the values of ref_idx_l0[mbPartIdx] shall be inferred to be equal to 0.

ref_idx_l1[mbPartIdx] has the same semantics as ref_idx_l0, with l0 and list 0 replaced by l1 and list 1, respectively.

mvd_l0[mbPartIdx][0][compIdx] specifies the difference between a vector component to be used and its prediction. The index mbPartIdx specifies to which macroblock partition mvd_l0 is assigned. The partitioning of the macroblock is specified by mb_type. The horizontal motion vector component difference is decoded first in decoding order and is assigned CompIdx = 0. The vertical motion vector component is decoded second in decoding order and is assigned CompIdx = 1. The range of the components of mvd_l0[mbPartIdx][0][compIdx] is specified by constraints on the motion vector variable values derived from it as specified in Annex A.

mvd_l1[mbPartIdx][0][compIdx] has the same semantics as mvd_l0, with l0 and L0 replaced by l1 and L1, respectively.

7.4.5.2 Sub-macroblock prediction semantics

sub_mb_type[mbPartIdx] specifies the sub-macroblock types.

Tables and semantics are specified for the various sub-macroblock types for P, and B macroblock types. Each table presents the value of sub_mb_type, the name of sub_mb_type, the number of sub-macroblock partitions used (given by the NumSubMbPart(sub_mb_type) function), and the prediction mode of the sub-macroblock (given by the SubMbPredMode(sub_mb_type) function). In the text, the value of sub_mb_type may be referred to by “sub-macroblock type”. In the text, the value of SubMbPredMode() may be referred to by “sub-macroblock prediction mode”.

The interpretation of sub_mb_type[mbPartIdx] for P macroblock types is specified in Table 7-17, where the row for “inferred” specifies values inferred when sub_mb_type[mbPartIdx] is not present.

Table 7-17 – Sub-macroblock types in P macroblocks

sub_mb_type[mbPartIdx]	Name of sub_mb_type[mbPartIdx]	NumSubMbPart (sub_mb_type[mbPartIdx])	SubMbPredMode (sub_mb_type[mbPartIdx])	SubMbPartWidth (sub_mb_type[mbPartIdx])	SubMbPartHeight (sub_mb_type[mbPartIdx])
inferred	na	na	na	na	na
0	P_L0_8x8	1	Pred_L0	8	8
1	P_L0_8x4	2	Pred_L0	8	4
2	P_L0_4x8	2	Pred_L0	4	8
3	P_L0_4x4	4	Pred_L0	4	4

The following semantics are assigned to the sub-macroblock types in Table 7-17.

- P_L0_MxN, with MxN being replaced by 8x8, 8x4, 4x8, or 4x4: the samples of the sub-macroblock are predicted using one luma partition of size MxN equal to 8x8, two luma partitions of size MxN equal to 8x4, or two luma partitions of size MxN equal to 4x8, or four luma partitions of size MxN equal to 4x4, and associated chroma samples, respectively.

The following semantics are assigned to the sub-macroblock prediction modes (SubMbPredMode()) in Table 7-17.

- Pred_L0: see semantics for Table 7-13.

The interpretation of `sub_mb_type[mbPartIdx]` for B macroblock types is specified in Table 7-18, where the row for "inferred" specifies values inferred when `sub_mb_type[mbPartIdx]` is not present, and the inferred value "mb_type" specifies that the name of `sub_mb_type[mbPartIdx]` is the same as the name of `mb_type` for this case.

Table 7-18 – Sub-macroblock types in B macroblocks

<code>sub_mb_type[mbPartIdx]</code>	Name of <code>sub_mb_type[mbPartIdx]</code>	NumSubMbPart (<code>sub_mb_type[mbPartIdx]</code>)	SubMbPredMode (<code>sub_mb_type[mbPartIdx]</code>)	SubMbPartWidth (<code>sub_mb_type[mbPartIdx]</code>)	SubMbPartHeight (<code>sub_mb_type[mbPartIdx]</code>)
inferred	mb_type	4	Direct	4	4
0	B_Direct_8x8	4	Direct	4	4
1	B_L0_8x8	1	Pred_L0	8	8
2	B_L1_8x8	1	Pred_L1	8	8
3	B_Bi_8x8	1	BiPred	8	8
4	B_L0_8x4	2	Pred_L0	8	4
5	B_L0_4x8	2	Pred_L0	4	8
6	B_L1_8x4	2	Pred_L1	8	4
7	B_L1_4x8	2	Pred_L1	4	8
8	B_Bi_8x4	2	BiPred	8	4
9	B_Bi_4x8	2	BiPred	4	8
10	B_L0_4x4	4	Pred_L0	4	4
11	B_L1_4x4	4	Pred_L1	4	4
12	B_Bi_4x4	4	BiPred	4	4

The following semantics are assigned to the sub-macroblock types in Table 7-18:

- B_Skip and B_Direct_16x16: no motion vector differences or reference indices are present for the sub-macroblock in the bitstream. The functions `SubMbPartWidth()` and `SubMbPartHeight()` are used in the derivation process for motion vectors and reference frame indices in subclause 8.4.1 for direct mode prediction.
- B_Direct_8x8: no motion vector differences or reference indices are present for the sub-macroblock in the bitstream. The functions `SubMbPartWidth(B_Direct_8x8)` and `SubMbPartHeight(B_Direct_8x8)` are used in the derivation process for motion vectors and reference frame indices in subclause 8.4.1 for direct mode prediction.
- B_X_MxN, with X being replaced by L0, L1, or Bi, and MxN being replaced by 8x8, 8x4, 4x8 or 4x4: the samples of the sub-macroblock are predicted using one luma partition of size MxN equal to 8x8, or the samples of the sub-macroblock are predicted using two luma partitions of size MxN equal to 8x4, or the samples of the sub-macroblock are predicted using two luma partitions of size MxN equal to 4x8, or the samples of the sub-macroblock are predicted using four luma partitions of size MxN equal to 4x4, and associated chroma samples, respectively. All sub-macroblock partitions share the same reference index. For an MxN sub-macroblock partition in a sub-macroblock with `sub_mb_type` being B_X_MxN with X being replaced by either L0 or L1, one motion vector difference is present in the bitstream. For an MxN sub-macroblock partition in a sub-macroblock with `sub_mb_type` being B_Bi_MxN, two motion vector difference are present in the bitstream.

The following semantics are assigned to the sub-macroblock prediction modes (SubMbPredMode()) in Table 7-18.

- Direct: see semantics for Table 7-14.
- Pred_L0: see semantics for Table 7-13.
- Pred_L1: see semantics for Table 7-14.
- BiPred: see semantics for Table 7-14.

ref_idx_l0[mbPartIdx] has the same semantics as ref_idx_l0 in subclause 7.4.5.1.

ref_idx_l1[mbPartIdx] has the same semantics as ref_idx_l1 in subclause 7.4.5.1.

mvd_l0[mbPartIdx][subMbPartIdx][compIdx] has the same semantics as mvd_l0 in subclause 7.4.5.1, except that it is applied to the sub-macroblock partition index with subMbPartIdx. The indices mbPartIdx and subMbPartIdx specify to which macroblock partition and sub-macroblock partition mvd_l0 is assigned.

mvd_l1[mbPartIdx][subMbPartIdx][compIdx] has the same semantics as mvd_l1 in subclause 7.4.5.1.

7.4.5.3 Residual data semantics

The syntax structure residual_block(), which is used for parsing the transform coefficient levels, is assigned as follows.

- If entropy_coding_mode_flag is equal to 0, residual_block is set equal to residual_block_cavlc, which is used for parsing the syntax elements for transform coefficient levels.
- Otherwise (entropy_coding_mode_flag is equal to 1), residual_block is set equal to residual_block_cabac, which is used for parsing the syntax elements for transform coefficient levels.

Depending on mb_type, luma or chroma, and chroma format, the syntax structure residual_block(coeffLevel, maxNumCoeff) is used with the arguments coeffLevel, which is a list containing the maxNumCoeff transform coefficient levels that are parsed in residual_block() and maxNumCoeff as follows.

- Depending on MbPartPredMode(mb_type, 0), the following applies.
 - If MbPartPredMode(mb_type, 0) is equal to Intra_16x16, the transform coefficient levels are parsed into the list Intra16x16DCLevel and into the 16 lists Intra16x16ACLevel[i]. Intra16x16DCLevel contains the 16 transform coefficient levels of the DC transform coefficient levels for each 4x4 luma block. For each of the 16 4x4 luma blocks indexed by i = 0..15, the 15 AC transform coefficients levels of the i-th block are parsed into the i-th list Intra16x16ACLevel[i].
 - Otherwise (MbPartPredMode(mb_type, 0) is not equal to Intra_16x16), the following applies.
 - If transform_size_8x8_flag is equal to 0, for each of the 16 4x4 luma blocks indexed by i = 0..15, the 16 transform coefficient levels of the i-th block are parsed into the i-th list LumaLevel[i].
 - Otherwise (transform_size_8x8_flag is equal to 1), for each of the 4 8x8 luma blocks indexed by i8x8 = 0..3, the following applies.
 - If entropy_coding_mode_flag is equal to 0, first for each of the 4 4x4 luma blocks indexed by i4x4 = 0..3, the 16 transform coefficient levels of the i4x4-th block are parsed into the (i8x8 * 4 + i4x4)-th list LumaLevel[i8x8 * 4 + i4x4]. Then, the 64 transform coefficient levels of the i8x8-th 8x8 luma block which are indexed by 4 * i + i4x4, where i = 0..15 and i4x4 = 0..3, are derived as LumaLevel8x8[i8x8][4 * i + i4x4] = LumaLevel[i8x8 * 4 + i4x4][i].
- NOTE – The 4x4 luma blocks with luma4x4BlkIdx = i8x8 * 4 + i4x4 containing every fourth transform coefficient level of the corresponding i8x8-th 8x8 luma block with offset i4x4 are assumed to represent spatial locations given by the inverse 4x4 luma block scanning process in subclause 6.4.3.
- Otherwise (entropy_coding_mode_flag is equal to 1), the 64 transform coefficient levels of the i8x8-th block are parsed into the i8x8-th list LumaLevel8x8[i8x8].
 - For each chroma component, indexed by iCbCr = 0..1, the DC transform coefficient levels of the 4 * NumC8x8 4x4 chroma blocks are parsed into the iCbCr-th list ChromaDCLevel[iCbCr].
 - For each of the 4x4 chroma blocks, indexed by i4x4 = 0..3 and i8x8 = 0..NumC8x8 – 1, of each chroma component, indexed by iCbCr = 0..1, the 15 AC transform coefficient levels are parsed into the (i8x8*4 + i4x4)-th list of the iCbCr-th chroma component ChromaACLevel[iCbCr][i8x8*4 + i4x4].

7.4.5.3.1 Residual block CAVLC semantics

The function `TotalCoeff(coeff_token)` that is used in subclause 7.3.5.3.1 returns the number of non-zero transform coefficient levels derived from `coeff_token`.

The function `TrailingOnes(coeff_token)` that is used in subclause 7.3.5.3.1 returns the trailing ones derived from `coeff_token`.

coeff_token specifies the total number of non-zero transform coefficient levels and the number of trailing one transform coefficient levels in a transform coefficient level scan. A trailing one transform coefficient level is one of up to three consecutive non-zero transform coefficient levels having an absolute value equal to 1 at the end of a scan of non-zero transform coefficient levels. The range of `coeff_token` is specified in subclause 9.2.1.

trailing_ones_sign_flag specifies the sign of a trailing one transform coefficient level as follows.

- If `trailing_ones_sign_flag` is equal to 0, the corresponding transform coefficient level is decoded as +1.
- Otherwise (`trailing_ones_sign_flag` equal to 1), the corresponding transform coefficient level is decoded as -1.

level_prefix and **level_suffix** specify the value of a non-zero transform coefficient level. The range of `level_prefix` and `level_suffix` is specified in subclause 9.2.2.

total_zeros specifies the total number of zero-valued transform coefficient levels that are located before the position of the last non-zero transform coefficient level in a scan of transform coefficient levels. The range of `total_zeros` is specified in subclause 9.2.3.

run_before specifies the number of consecutive transform coefficient levels in the scan with zero value before a non-zero valued transform coefficient level. The range of `run_before` is specified in subclause 9.2.3.

`coeffLevel` contains `maxNumCoeff` transform coefficient levels for the current list of transform coefficient levels.

7.4.5.3.2 Residual block CABAC semantics

coded_block_flag specifies whether the block contains non-zero transform coefficient levels as follows.

- If `coded_block_flag` is equal to 0, the block contains no non-zero transform coefficient levels.
- Otherwise (`coded_block_flag` is equal to 1), the block contains at least one non-zero transform coefficient level.

significant_coeff_flag[i] specifies whether the transform coefficient level at scanning position `i` is non-zero as follows.

- If `significant_coeff_flag[i]` is equal to 0, the transform coefficient level at scanning position `i` is set equal to 0;
- Otherwise (`significant_coeff_flag[i]` is equal to 1), the transform coefficient level at scanning position `i` has a non-zero value.

last_significant_coeff_flag[i] specifies for the scanning position `i` whether there are non-zero transform coefficient levels for subsequent scanning positions `i + 1` to `maxNumCoeff - 1` as follows.

- If `last_significant_coeff_flag[i]` is equal to 1, all following transform coefficient levels (in scanning order) of the block have value equal to 0.
- Otherwise (`last_significant_coeff_flag[i]` is equal to 0), there are further non-zero transform coefficient levels along the scanning path.

coeff_abs_level_minus1[i] is the absolute value of a transform coefficient level minus 1. The value of `coeff_abs_level_minus1` is constrained by the limits in subclause 8.5.

coeff_sign_flag[i] specifies the sign of a transform coefficient level as follows.

- If `coeff_sign_flag` is equal to 0, the corresponding transform coefficient level has a positive value.
- Otherwise (`coeff_sign_flag` is equal to 1), the corresponding transform coefficient level has a negative value.

`coeffLevel` contains `maxNumCoeff` transform coefficient levels for the current list of transform coefficient levels.

8 Decoding process

Outputs of this process are decoded samples of the current picture (sometimes referred to by the variable `CurrPic`).

This clause describes the decoding process, given syntax elements and upper-case variables from clause 7.

The decoding process is specified such that all decoders shall produce numerically identical results. Any decoding process that produces identical results to the process described here conforms to the decoding process requirements of this Recommendation | International Standard.

Each picture referred to in this clause is a primary picture. Each slice referred to in this clause is a slice of a primary picture. Each slice data partition referred to in this clause is a slice data partition of a primary picture.

An overview of the decoding process is given as follows.

- The decoding of NAL units is specified in subclause 8.1.
- The processes in subclause 8.2 specify decoding processes using syntax elements in the slice layer and above.
 - Variables and functions relating to picture order count are derived in subclause 8.2.1. (only needed to be invoked for one slice of a picture)
 - Variables and functions relating to the macroblock to slice group map are derived in subclause 8.2.2. (only needed to be invoked for one slice of a picture)
 - The method of combining the various partitions when slice data partitioning is used is described in subclause 8.2.3.
 - When the frame_num of the current picture is not equal to PrevRefFrameNum and is not equal to $(\text{PrevRefFrameNum} + 1) \% \text{MaxFrameNum}$, the decoding process for gaps in frame_num is performed according to subclause 8.2.5.2 prior to the decoding of any slices of the current picture.
 - At the beginning of the decoding process for each P, SP, or B slice, the decoding process for reference picture lists construction specified in 8.2.4 performed for derivation of reference picture list 0 (RefPicList0), and when decoding a B slice, reference picture list 1 (RefPicList1).
 - When the current picture is a reference picture and after all slices of the current picture have been decoded, the decoded reference picture marking process in subclause 8.2.5 specifies how the current picture is used in the decoding process of inter prediction in later decoded pictures.
- The processes in subclauses 8.3, 8.4, 8.5, 8.6, and 8.7 specify decoding processes using syntax elements in the macroblock layer and above.
 - The intra prediction process for I and SI macroblocks, except for I_PCM macroblocks as specified in subclause 8.3, has intra prediction samples as its output. For I_PCM macroblocks subclause 8.3 directly specifies a picture construction process. The output are the constructed samples prior to the deblocking filter process.
 - The inter prediction process for P and B macroblocks is specified in subclause 8.4 with inter prediction samples being the output.
 - The transform coefficient decoding process and picture construction process prior to deblocking filter process are specified in subclause 8.5. That process derives samples for I and B macroblocks and for P macroblocks in P slices. The output are constructed samples prior to the deblocking filter process.
 - The decoding process for P macroblocks in SP slices or SI macroblocks is specified in subclause 8.6. That process derives samples for P macroblocks in SP slices and for SI macroblocks. The output are constructed samples prior to the deblocking filter process.
 - The constructed samples prior to the deblocking filter process that are next to the edges of blocks and macroblocks are processed by a deblocking filter as specified in subclause 8.7 with the output being the decoded samples.

8.1 NAL unit decoding process

Inputs to this process are NAL units.

Outputs of this process are the RBSP syntax structures encapsulated within the NAL units.

The decoding process for each NAL unit extracts the RBSP syntax structure from the NAL unit and then operates the decoding processes specified for the RBSP syntax structure in the NAL unit as follows.

Subclause 8.2 describes the decoding process for NAL units with nal_unit_type equal to 1 through 5.

Subclauses 8.3 describes the decoding process for a macroblock or part of a macroblock coded in NAL units with nal_unit_type equal to 1, 2, and 5.

Subclause 8.4 describes the decoding process for a macroblock or part of a macroblock coded in NAL units with nal_unit_type equal to 1 and 2.

Subclause 8.5 describes the decoding process for a macroblock or part of a macroblock coded in NAL units with nal_unit_type equal to 1 and 3 to 5.

Subclause 8.6 describes the decoding process for a macroblock or part of a macroblock coded in NAL units with nal_unit_type equal to 1 and 3 to 5.

Subclause 8.7 describes the decoding process for a macroblock or part of a macroblock coded in NAL units with nal_unit_type equal to 1 to 5.

NAL units with nal_unit_type equal to 7 and 8 contain sequence parameter sets and picture parameter sets, respectively. Picture parameter sets are used in the decoding processes of other NAL units as determined by reference to a picture parameter set within the slice headers of each picture. Sequence parameter sets are used in the decoding processes of other NAL units as determined by reference to a sequence parameter set within the picture parameter sets of each sequence.

No normative decoding process is specified for NAL units with nal_unit_type equal to 6, 9, 10, 11, and 12.

8.2 Slice decoding process

8.2.1 Decoding process for picture order count

Outputs of this process are TopFieldOrderCnt (if applicable) and BottomFieldOrderCnt (if applicable).

Picture order counts are used to determine initial picture orderings for reference pictures in the decoding of B slices (see subclauses 8.2.4.2.3 and 8.2.4.2.4), to represent picture order differences between frames or fields for motion vector derivation in temporal direct mode (see subclause 8.4.1.2.3), for implicit mode weighted prediction in B slices (see subclause 8.4.2.3.2), and for decoder conformance checking (see subclause C.4).

Picture order count information is derived for every frame, field (whether decoded from a coded field or as a part of a decoded frame), or complementary field pair as follows:

- Each coded frame is associated with two picture order counts, called TopFieldOrderCnt and BottomFieldOrderCnt for its top field and bottom field, respectively.
- Each coded field is associated with a picture order count, called TopFieldOrderCnt for a coded top field and BottomFieldOrderCnt for a bottom field.
- Each complementary field pair is associated with two picture order counts, which are the TopFieldOrderCnt for its coded top field and the BottomFieldOrderCnt for its coded bottom field, respectively.

TopFieldOrderCnt and BottomFieldOrderCnt indicate the picture order of the corresponding top field or bottom field relative to the first output field of the previous IDR picture or the previous reference picture including a memory_management_control_operation equal to 5 in decoding order.

TopFieldOrderCnt and BottomFieldOrderCnt are derived by invoking one of the decoding processes for picture order count type 0, 1, and 2 in subclauses 8.2.1.1, 8.2.1.2, and 8.2.1.3, respectively. When the current picture includes a memory management control operation equal to 5, after the decoding of the current picture, tempPicOrderCnt is set equal to PicOrderCnt(CurrPic), TopFieldOrderCnt of the current picture (if any) is set equal to TopFieldOrderCnt - tempPicOrderCnt, and BottomFieldOrderCnt of the current picture (if any) is set equal to BottomFieldOrderCnt - tempPicOrderCnt.

The bitstream shall not contain data that results in $\text{Min}(\text{TopFieldOrderCnt}, \text{BottomFieldOrderCnt})$ not equal to 0 for a coded IDR frame, TopFieldOrderCnt not equal to 0 for a coded IDR top field, or BottomFieldOrderCnt not equal to 0 for a coded IDR bottom field. Thus, at least one of TopFieldOrderCnt and BottomFieldOrderCnt shall be equal to 0 for the fields of a coded IDR frame.

When the current picture is not an IDR picture, the following applies.

- Consider the list variable listD containing as elements the TopFieldOrderCnt and BottomFieldOrderCnt values associated with the list of pictures including all of the following
 - the first picture in the list is the previous picture of any of the following types
 - an IDR picture
 - a picture containing a memory_management_control_operation equal to 5
 - the following additional pictures.

- If `pic_order_cnt_type` is equal to 0, all other pictures that follow in decoding order after the first picture in the list and are not "non-existing" frames inferred by the decoding process for gaps in `frame_num` specified in subclause 8.2.5.2 and either precede the current picture in decoding order or are the current picture. When `pic_order_cnt_type` is equal to 0 and the current picture is not a "non-existing" frame inferred by the decoding process for gaps in `frame_num` specified in subclause 8.2.5.2, the current picture is included in `listD` prior to the invoking of the decoded reference picture marking process.

Otherwise (`pic_order_cnt_type` is not equal to 0), all other pictures that follow in decoding order after the first picture in the list and either precede the current picture in decoding order or are the current picture. When `pic_order_cnt_type` is not equal to 0, the current picture is included in `listD` prior to the invoking of the decoded reference picture marking process.

- Consider the list variable `listO` which contains the elements of `listD` sorted in ascending order. `listO` shall not contain any of the following.
 - a pair of `TopFieldOrderCnt` and `BottomFieldOrderCnt` for a frame or complementary field pair that are not at consecutive positions in `listO`.
 - a `TopFieldOrderCnt` that has a value equal to another `TopFieldOrderCnt`.
 - a `BottomFieldOrderCnt` that has a value equal to another `BottomFieldOrderCnt`.
 - a `BottomFieldOrderCnt` that has a value equal to a `TopFieldOrderCnt` unless the `BottomFieldOrderCnt` and `TopFieldOrderCnt` belong to the same coded frame or complementary field pair.

The bitstream shall not contain data that results in values of `TopFieldOrderCnt`, `BottomFieldOrderCnt`, `PicOrderCntMsb`, or `FrameNumOffset` used in the decoding process as specified in subclauses 8.2.1.1 to 8.2.1.3 that exceed the range of values from -2^{31} to $2^{31}-1$, inclusive.

The function `PicOrderCnt(picX)` is specified as follows:

```

if( picX is a frame or a complementary field pair )
    PicOrderCnt( picX ) = Min( TopFieldOrderCnt, BottomFieldOrderCnt ) of the frame or complementary field
    pair picX
else if( picX is a top field )
    PicOrderCnt( picX ) = TopFieldOrderCnt of field picX
else if( picX is a bottom field )
    PicOrderCnt( picX ) = BottomFieldOrderCnt of field picX

```

(8-1)

Then `DiffPicOrderCnt(picA, picB)` is specified as follows:

$$\text{DiffPicOrderCnt}(\text{picA}, \text{picB}) = \text{PicOrderCnt}(\text{picA}) - \text{PicOrderCnt}(\text{picB})$$

(8-2)

The bitstream shall not contain data that results in values of `DiffPicOrderCnt(picA, picB)` used in the decoding process that exceed the range of -2^{15} to $2^{15} - 1$, inclusive.

NOTE 1 – Let *X* be the current picture and *Y* and *Z* be two other pictures in the same sequence, *Y* and *Z* are considered to be in the same output order direction from *X* when both `DiffPicOrderCnt(X, Y)` and `DiffPicOrderCnt(X, Z)` are positive or both are negative.

NOTE 2 – Many applications assign `PicOrderCnt(X)` proportional to the sampling time of the picture *X* relative to the sampling time of an IDR picture.

When the current picture includes a `memory_management_control_operation` equal to 5, `PicOrderCnt(CurrPic)` shall be greater than `PicOrderCnt(any other picture in listD)`.

8.2.1.1 Decoding process for picture order count type 0

This process is invoked when `pic_order_cnt_type` is equal to 0.

Input to this process is `PicOrderCntMsb` of the previous reference picture in decoding order as specified in this subclause.

Outputs of this process are either or both `TopFieldOrderCnt` or `BottomFieldOrderCnt`.

The variables `prevPicOrderCntMsb` and `prevPicOrderCntLsb` are derived as follows.

- If the current picture is an IDR picture, `prevPicOrderCntMsb` is set equal to 0 and `prevPicOrderCntLsb` is set equal to 0.
- Otherwise (the current picture is not an IDR picture), the following applies.

- If the previous reference picture in decoding order included a memory_management_control_operation equal to 5, the following applies.
 - If the previous reference picture in decoding order is not a bottom field, prevPicOrderCntMsb is set equal to 0 and prevPicOrderCntLsb is set equal to the value of TopFieldOrderCnt for the previous reference picture in decoding order.
 - Otherwise (the previous reference picture in decoding order is a bottom field), prevPicOrderCntMsb is set equal to 0 and prevPicOrderCntLsb is set equal to 0.
- Otherwise (the previous reference picture in decoding order did not include a memory_management_control_operation equal to 5), prevPicOrderCntMsb is set equal to PicOrderCntMsb of the previous reference picture in decoding order and prevPicOrderCntLsb is set equal to the value of pic_order_cnt_lsb of the previous reference picture in decoding order.

PicOrderCntMsb of the current picture is derived as follows:

```

if( ( pic_order_cnt_lsb < prevPicOrderCntLsb ) &&
    ( ( prevPicOrderCntLsb - pic_order_cnt_lsb ) >= ( MaxPicOrderCntLsb / 2 ) ) )
    PicOrderCntMsb = prevPicOrderCntMsb + MaxPicOrderCntLsb
else if( ( pic_order_cnt_lsb > prevPicOrderCntLsb ) &&
    ( ( pic_order_cnt_lsb - prevPicOrderCntLsb ) > ( MaxPicOrderCntLsb / 2 ) ) )
    PicOrderCntMsb = prevPicOrderCntMsb - MaxPicOrderCntLsb
else
    PicOrderCntMsb = prevPicOrderCntMsb

```

(8-3)

When the current picture is not a bottom field, TopFieldOrderCnt is derived as follows:

```

if( !field_pic_flag || !bottom_field_flag )
    TopFieldOrderCnt = PicOrderCntMsb + pic_order_cnt_lsb

```

(8-4)

When the current picture is not a top field, BottomFieldOrderCnt is derived as follows:

```

if( !field_pic_flag )
    BottomFieldOrderCnt = TopFieldOrderCnt + delta_pic_order_cnt_bottom
else if( bottom_field_flag )
    BottomFieldOrderCnt = PicOrderCntMsb + pic_order_cnt_lsb

```

(8-5)

8.2.1.2 Decoding process for picture order count type 1

This process is invoked when pic_order_cnt_type is equal to 1.

Input to this process is FrameNumOffset of the previous picture in decoding order as specified in this subclause.

Outputs of this process are either or both TopFieldOrderCnt or BottomFieldOrderCnt.

The values of TopFieldOrderCnt and BottomFieldOrderCnt are derived as specified in this subclause. Let prevFrameNum be equal to the frame_num of the previous picture in decoding order.

When the current picture is not an IDR picture, the variable prevFrameNumOffset is derived as follows.

- If the previous picture in decoding order included a memory_management_control_operation equal to 5, prevFrameNumOffset is set equal to 0.
 - Otherwise (the previous picture in decoding order did not include a memory_management_control_operation equal to 5), prevFrameNumOffset is set equal to the value of FrameNumOffset of the previous picture in decoding order.
- NOTE – When gaps_in_frame_num_value_allowed_flag is equal to 1, the previous picture in decoding order may be a "non-existing" frame inferred by the decoding process for gaps in frame_num specified in subclause 8.2.5.2.

The derivation proceeds in the following ordered steps.

1. The variable FrameNumOffset is derived as follows:

```

if( nal_unit_type == 5 )
    FrameNumOffset = 0
else if( prevFrameNum > frame_num )
    FrameNumOffset = prevFrameNumOffset + MaxFrameNum

```

(8-6)

```

else
    FrameNumOffset = prevFrameNumOffset

```

2. The variable `absFrameNum` is derived as follows:

```

if( num_ref_frames_in_pic_order_cnt_cycle != 0 )
    absFrameNum = FrameNumOffset + frame_num
else
    absFrameNum = 0
if( nal_ref_idc == 0 && absFrameNum > 0 )
    absFrameNum = absFrameNum - 1

```

(8-7)

3. When `absFrameNum > 0`, `picOrderCntCycleCnt` and `frameNumInPicOrderCntCycle` are derived as follows:

```

if( absFrameNum > 0 ) {
    picOrderCntCycleCnt = ( absFrameNum - 1 ) / num_ref_frames_in_pic_order_cnt_cycle
    frameNumInPicOrderCntCycle = ( absFrameNum - 1 ) % num_ref_frames_in_pic_order_cnt_cycle
}

```

(8-8)

4. The variable `expectedDeltaPerPicOrderCntCycle` is derived as follows:

```

expectedDeltaPerPicOrderCntCycle = 0
for( i = 0; i < num_ref_frames_in_pic_order_cnt_cycle; i++ )
    expectedDeltaPerPicOrderCntCycle += offset_for_ref_frame[ i ]

```

(8-9)

5. The variable `expectedPicOrderCnt` is derived as follows:

```

if( absFrameNum > 0 ){
    expectedPicOrderCnt = picOrderCntCycleCnt * expectedDeltaPerPicOrderCntCycle
    for( i = 0; i <= frameNumInPicOrderCntCycle; i++ )
        expectedPicOrderCnt = expectedPicOrderCnt + offset_for_ref_frame[ i ]
} else
    expectedPicOrderCnt = 0
if( nal_ref_idc == 0 )
    expectedPicOrderCnt = expectedPicOrderCnt + offset_for_non_ref_pic

```

(8-10)

6. The variables `TopFieldOrderCnt` or `BottomFieldOrderCnt` are derived as follows:

```

if( !field_pic_flag ) {
    TopFieldOrderCnt = expectedPicOrderCnt + delta_pic_order_cnt[ 0 ]
    BottomFieldOrderCnt = TopFieldOrderCnt +
        offset_for_top_to_bottom_field + delta_pic_order_cnt[ 1 ]
} else if( !bottom_field_flag )
    TopFieldOrderCnt = expectedPicOrderCnt + delta_pic_order_cnt[ 0 ]
else
    BottomFieldOrderCnt = expectedPicOrderCnt + offset_for_top_to_bottom_field + delta_pic_order_cnt[ 0 ]

```

(8-11)

8.2.1.3 Decoding process for picture order count type 2

This process is invoked when `pic_order_cnt_type` is equal to 2.

Outputs of this process are either or both `TopFieldOrderCnt` or `BottomFieldOrderCnt`.

Let `prevFrameNum` be equal to the `frame_num` of the previous picture in decoding order.

When the current picture is not an IDR picture, the variable `prevFrameNumOffset` is derived as follows.

- If the previous picture in decoding order included a `memory_management_control_operation` equal to 5, `prevFrameNumOffset` is set equal to 0.
- Otherwise (the previous picture in decoding order did not include a `memory_management_control_operation` equal to 5), `prevFrameNumOffset` is set equal to the value of `FrameNumOffset` of the previous picture in decoding order.

NOTE 1 – When `gaps_in_frame_num_value_allowed_flag` is equal to 1, the previous picture in decoding order may be a "non-existing" frame inferred by the decoding process for gaps in `frame_num` specified in subclause 8.2.5.2.

The variable `FrameNumOffset` is derived as follows.

```

if( nal_unit_type == 5 )
    FrameNumOffset = 0
else if( prevFrameNum > frame_num )
    FrameNumOffset = prevFrameNumOffset + MaxFrameNum
else
    FrameNumOffset = prevFrameNumOffset

```

(8-12)

The variable `tempPicOrderCnt` is derived as follows:

```

if( nal_unit_type == 5 )
    tempPicOrderCnt = 0
else if( nal_ref_idc == 0 )
    tempPicOrderCnt = 2 * ( FrameNumOffset + frame_num ) - 1
else
    tempPicOrderCnt = 2 * ( FrameNumOffset + frame_num )

```

(8-13)

The variables `TopFieldOrderCnt` or `BottomFieldOrderCnt` are derived as follows:

```

if( !field_pic_flag ) {
    TopFieldOrderCnt = tempPicOrderCnt
    BottomFieldOrderCnt = tempPicOrderCnt
} else if( bottom_field_flag )
    BottomFieldOrderCnt = tempPicOrderCnt
else
    TopFieldOrderCnt = tempPicOrderCnt

```

(8-14)

NOTE 2 – Picture order count type 2 cannot be used in a coded video sequence that contains consecutive non-reference pictures that would result in more than one of these pictures having the same value of `TopFieldOrderCnt` or more than one of these pictures having the same value of `BottomFieldOrderCnt`.

NOTE 3 – Picture order count type 2 results in an output order that is the same as the decoding order.

8.2.2 Decoding process for macroblock to slice group map

Inputs to this process are the active picture parameter set and the slice header of the slice to be decoded.

Output of this process is a macroblock to slice group map `MbToSliceGroupMap`.

This process is invoked at the start of every slice.

NOTE – The output of this process is equal for all slices of a picture.

When `num_slice_groups_minus1` is equal to 1 and `slice_group_map_type` is equal to 3, 4, or 5, slice groups 0 and 1 have a size and shape determined by `slice_group_change_direction_flag` as shown in Table 8-1 and specified in subclauses 8.2.2.4 to 8.2.2.6.

Table 8-1 – Refined slice group map type

slice_group_map_type	slice_group_change_direction_flag	refined slice group map type
3	0	Box-out clockwise
3	1	Box-out counter-clockwise
4	0	Raster scan
4	1	Reverse raster scan
5	0	Wipe right
5	1	Wipe left

In such a case, MapUnitsInSliceGroup0 slice group map units in the specified growth order are allocated for slice group 0 and the remaining PicSizeInMapUnits – MapUnitsInSliceGroup0 slice group map units of the picture are allocated for slice group 1.

When num_slice_groups_minus1 is equal to 1 and slice_group_map_type is equal to 4 or 5, the variable sizeOfUpperLeftGroup is defined as follows:

$$\text{sizeOfUpperLeftGroup} = (\text{slice_group_change_direction_flag} ? (\text{PicSizeInMapUnits} - \text{MapUnitsInSliceGroup0}) : \text{MapUnitsInSliceGroup0}) \quad (8-15)$$

The variable mapUnitToSliceGroupMap is derived as follows.

- If num_slice_groups_minus1 is equal to 0, the map unit to slice group map is generated for all i ranging from 0 to PicSizeInMapUnits – 1, inclusive, as specified by:

$$\text{mapUnitToSliceGroupMap}[i] = 0 \quad (8-16)$$

- Otherwise (num_slice_groups_minus1 is not equal to 0), mapUnitToSliceGroupMap is derived as follows.
 - If slice_group_map_type is equal to 0, the derivation of mapUnitToSliceGroupMap as specified in subclause 8.2.2.1 applies.
 - Otherwise, if slice_group_map_type is equal to 1, the derivation of mapUnitToSliceGroupMap as specified in subclause 8.2.2.2 applies.
 - Otherwise, if slice_group_map_type is equal to 2, the derivation of mapUnitToSliceGroupMap as specified in subclause 8.2.2.3 applies.
 - Otherwise, if slice_group_map_type is equal to 3, the derivation of mapUnitToSliceGroupMap as specified in subclause 8.2.2.4 applies.
 - Otherwise, if slice_group_map_type is equal to 4, the derivation of mapUnitToSliceGroupMap as specified in subclause 8.2.2.5 applies.
 - Otherwise, if slice_group_map_type is equal to 5, the derivation of mapUnitToSliceGroupMap as specified in subclause 8.2.2.6 applies.
 - Otherwise (slice_group_map_type is equal to 6), the derivation of mapUnitToSliceGroupMap as specified in subclause 8.2.2.7 applies.

After derivation of the mapUnitToSliceGroupMap, the process specified in subclause 8.2.2.8 is invoked to convert the map unit to slice group map mapUnitToSliceGroupMap to the macroblock to slice group map MbToSliceGroupMap. After derivation of the macroblock to slice group map as specified in subclause 8.2.2.8, the function NextMbAddress(n) is defined as the value of the variable nextMbAddress derived as specified by:

$$\begin{aligned} i &= n + 1 \\ \text{while}(i < \text{PicSizeInMbs} \ \&\& \ \text{MbToSliceGroupMap}[i] \neq \text{MbToSliceGroupMap}[n]) \\ &\quad i++; \\ \text{nextMbAddress} &= i \end{aligned} \quad (8-17)$$

8.2.2.1 Specification for interleaved slice group map type

The specifications in this subclause apply when slice_group_map_type is equal to 0.

The map unit to slice group map is generated as specified by:

$$\begin{aligned} i &= 0 \\ \text{do} \\ &\quad \text{for}(iGroup = 0; iGroup \leq \text{num_slice_groups_minus1} \ \&\& \ i < \text{PicSizeInMapUnits}; \\ &\quad \quad i += \text{run_length_minus1}[iGroup++] + 1) \\ &\quad \quad \text{for}(j = 0; j \leq \text{run_length_minus1}[iGroup] \ \&\& \ i + j < \text{PicSizeInMapUnits}; j++) \\ &\quad \quad \quad \text{mapUnitToSliceGroupMap}[i + j] = iGroup \\ \text{while}(i < \text{PicSizeInMapUnits}) \end{aligned} \quad (8-18)$$

8.2.2.2 Specification for dispersed slice group map type

The specifications in this subclause apply when slice_group_map_type is equal to 1.

The map unit to slice group map is generated as specified by:

```
for( i = 0; i < PicSizeInMapUnits; i++ )
    mapUnitToSliceGroupMap[ i ] = ( ( i % PicWidthInMbs ) +
        ( ( i / PicWidthInMbs ) * ( num_slice_groups_minus1 + 1 ) ) / 2 )
        % ( num_slice_groups_minus1 + 1 )
```

(8-19)

8.2.2.3 Specification for foreground with left-over slice group map type

The specifications in this subclause apply when slice_group_map_type is equal to 2.

The map unit to slice group map is generated as specified by:

```
for( i = 0; i < PicSizeInMapUnits; i++ )
    mapUnitToSliceGroupMap[ i ] = num_slice_groups_minus1
for( iGroup = num_slice_groups_minus1 - 1; iGroup >= 0; iGroup-- ) {
    yTopLeft = top_left[ iGroup ] / PicWidthInMbs
    xTopLeft = top_left[ iGroup ] % PicWidthInMbs
    yBottomRight = bottom_right[ iGroup ] / PicWidthInMbs
    xBottomRight = bottom_right[ iGroup ] % PicWidthInMbs
    for( y = yTopLeft; y <= yBottomRight; y++ )
        for( x = xTopLeft; x <= xBottomRight; x++ )
            mapUnitToSliceGroupMap[ y * PicWidthInMbs + x ] = iGroup
}
```

(8-20)

NOTE – The rectangles may overlap. Slice group 0 contains the macroblocks that are within the rectangle specified by top_left[0] and bottom_right[0]. A slice group having slice group ID greater than 0 and less than num_slice_groups_minus1 contains the macroblocks that are within the specified rectangle for that slice group that are not within the rectangle specified for any slice group having a smaller slice group ID. The slice group with slice group ID equal to num_slice_groups_minus1 contains the macroblocks that are not in the other slice groups.

8.2.2.4 Specification for box-out slice group map types

The specifications in this subclause apply when slice_group_map_type is equal to 3.

The map unit to slice group map is generated as specified by:

```
for( i = 0; i < PicSizeInMapUnits; i++ )
    mapUnitToSliceGroupMap[ i ] = 1
x = ( PicWidthInMbs - slice_group_change_direction_flag ) / 2
y = ( PicHeightInMapUnits - slice_group_change_direction_flag ) / 2
( leftBound, topBound ) = ( x, y )
( rightBound, bottomBound ) = ( x, y )
( xDir, yDir ) = ( slice_group_change_direction_flag - 1, slice_group_change_direction_flag )
for( k = 0; k < MapUnitsInSliceGroup0; k += mapUnitVacant ) {
    mapUnitVacant = ( mapUnitToSliceGroupMap[ y * PicWidthInMbs + x ] == 1 )
    if( mapUnitVacant )
        mapUnitToSliceGroupMap[ y * PicWidthInMbs + x ] = 0
    if( xDir == -1 && x == leftBound ) {
        leftBound = Max( leftBound - 1, 0 )
        x = leftBound
        ( xDir, yDir ) = ( 0, 2 * slice_group_change_direction_flag - 1 )
    } else if( xDir == 1 && x == rightBound ) {
        rightBound = Min( rightBound + 1, PicWidthInMbs - 1 )
        x = rightBound
        ( xDir, yDir ) = ( 0, 1 - 2 * slice_group_change_direction_flag )
    } else if( yDir == -1 && y == topBound ) {
        topBound = Max( topBound - 1, 0 )
        y = topBound
        ( xDir, yDir ) = ( 1 - 2 * slice_group_change_direction_flag, 0 )
    } else if( yDir == 1 && y == bottomBound ) {
        bottomBound = Min( bottomBound + 1, PicHeightInMapUnits - 1 )
        y = bottomBound
        ( xDir, yDir ) = ( 2 * slice_group_change_direction_flag - 1, 0 )
    } else
```

(8-21)

```

        ( x, y ) = ( x + xDir, y + yDir )
    }

```

8.2.2.5 Specification for raster scan slice group map types

The specifications in this subclause apply when slice_group_map_type is equal to 4.

The map unit to slice group map is generated as specified by:

```

for( i = 0; i < PicSizeInMapUnits; i++ )
    if( i < sizeOfUpperLeftGroup )
        mapUnitToSliceGroupMap[ i ] = slice_group_change_direction_flag
    else
        mapUnitToSliceGroupMap[ i ] = 1 - slice_group_change_direction_flag

```

(8-22)

8.2.2.6 Specification for wipe slice group map types

The specifications in this subclause apply when slice_group_map_type is equal to 5.

The map unit to slice group map is generated as specified by:

```

k = 0;
for( j = 0; j < PicWidthInMbs; j++ )
    for( i = 0; i < PicHeightInMapUnits; i++ )
        if( k++ < sizeOfUpperLeftGroup )
            mapUnitToSliceGroupMap[ i * PicWidthInMbs + j ] = slice_group_change_direction_flag
        else
            mapUnitToSliceGroupMap[ i * PicWidthInMbs + j ] = 1 - slice_group_change_direction_flag

```

(8-23)

8.2.2.7 Specification for explicit slice group map type

The specifications in this subclause apply when slice_group_map_type is equal to 6.

The map unit to slice group map is generated as specified by:

$$\text{mapUnitToSliceGroupMap}[i] = \text{slice_group_id}[i] \quad (8-24)$$

for all i ranging from 0 to PicSizeInMapUnits – 1, inclusive.

8.2.2.8 Specification for conversion of map unit to slice group map to macroblock to slice group map

For each value of i ranging from 0 to PicSizeInMbs – 1, inclusive, the macroblock to slice group map is specified as follows.

- If frame_mbs_only_flag is equal to 1 or field_pic_flag is equal to 1, the macroblock to slice group map is specified by:

$$\text{MbToSliceGroupMap}[i] = \text{mapUnitToSliceGroupMap}[i] \quad (8-25)$$

- Otherwise, if MbaffFrameFlag is equal to 1, the macroblock to slice group map is specified by:

$$\text{MbToSliceGroupMap}[i] = \text{mapUnitToSliceGroupMap}[i / 2] \quad (8-26)$$

- Otherwise (frame_mbs_only_flag is equal to 0 and mb_adaptive_frame_field_flag is equal to 0 and field_pic_flag is equal to 0), the macroblock to slice group map is specified by:

$$\begin{aligned} \text{MbToSliceGroupMap}[i] = \text{mapUnitToSliceGroupMap}[(i / (2 * \text{PicWidthInMbs})) * \text{PicWidthInMbs} \\ + (i \% \text{PicWidthInMbs})] \end{aligned} \quad (8-27)$$

8.2.3 Decoding process for slice data partitioning

Inputs to this process are

- a slice data partition A layer RBSP,

- when syntax elements of category 3 are present in the slice data, a slice data partition B layer RBSP having the same slice_id as in the slice data partition A layer RBSP, and
- when syntax elements of category 4 are present in the slice data, a slice data partition C layer RBSP having the same slice_id as in the slice data partition A layer RBSP.

NOTE 1 – The slice data partition B layer RBSP and slice data partition C layer RBSP need not be present.

Output of this process is a coded slice.

When slice data partitioning is not used, coded slices are represented by a slice layer without partitioning RBSP that contains a slice header followed by a slice data syntax structure that contains all the syntax elements of categories 2, 3, and 4 (see category column in subclause 7.3) of the macroblock data for the macroblocks of the slice.

When slice data partitioning is used, the macroblock data of a slice is partitioned into one to three partitions contained in separate NAL units. Partition A contains a slice data partition A header, and all syntax elements of category 2. Partition B, when present, contains a slice data partition B header and all syntax elements of category 3. Partition C, when present, contains a slice data partition C header and all syntax elements of category 4.

When slice data partitioning is used, the syntax elements of each category are parsed from a separate NAL unit, which need not be present when no symbols of the respective category exist. The decoding process shall process the slice data partitions of a coded slice in a manner equivalent to processing a corresponding slice layer without partitioning RBSP by extracting each syntax element from the slice data partition in which the syntax element appears depending on the slice data partition assignment in the syntax tables in subclause 7.3.

NOTE 2 – Syntax elements of category 3 are relevant to the decoding of residual data of I and SI macroblock types. Syntax elements of category 4 are relevant to the decoding of residual data of P and B macroblock types. Category 2 encompasses all other syntax elements related to the decoding of macroblocks, and their information is often denoted as header information. The slice data partition A header contains all the syntax elements of the slice header, and additionally a slice_id that are used to associate the slice data partitions B and C with the slice data partition A. The slice data partition B and C headers contain the slice_id syntax element that establishes their association with the slice data partition A of the slice.

8.2.4 Decoding process for reference picture lists construction

This process is invoked at the beginning of decoding of each P, SP, or B slice.

Decoded reference pictures are marked as "used for short-term reference" or "used for long-term reference" as specified by the bitstream and specified in subclause 8.2.5. Short-term reference pictures are identified by the value of frame_num. Long-term reference pictures are assigned a long-term frame index as specified by the bitstream and specified in subclause 8.2.5.

Subclause 8.2.4.1 is invoked to specify

- the assignment of variables FrameNum, FrameNumWrap, and PicNum to each of the short-term reference pictures, and
- the assignment of variable LongTermPicNum to each of the long-term reference pictures.

Reference pictures are addressed through reference indices as specified in subclause 8.4.2.1. A reference index is an index into a reference picture list. When decoding a P or SP slice, there is a single reference picture list RefPicList0. When decoding a B slice, there is a second independent reference picture list RefPicList1 in addition to RefPicList0.

At the beginning of decoding of each slice, reference picture list RefPicList0, and for B slices RefPicList1, are derived as follows.

- An initial reference picture list RefPicList0 and for B slices RefPicList1 are derived as specified in subclause 8.2.4.2.
- The initial reference picture list RefPicList0 and for B slices RefPicList1 are modified as specified in subclause 8.2.4.3.

NOTE – The reordering process for reference picture lists specified in subclause 8.2.4.3 allows the contents of RefPicList0 and for B slices RefPicList1 to be modified in a flexible fashion. In particular, it is possible for a picture that is currently marked "used for reference" to be inserted into RefPicList0 and for B slices RefPicList1 even when the picture is not in the initial reference picture list derived as specified in subclause 8.2.4.2.

The number of entries in the modified reference picture list RefPicList0 is num_ref_idx_l0_active_minus1 + 1, and for B slices the number of entries in the modified reference picture list RefPicList1 is num_ref_idx_l1_active_minus1 + 1. A reference picture may appear at more than one index in the modified reference picture lists RefPicList0 or RefPicList1.

8.2.4.1 Decoding process for picture numbers

This process is invoked when the decoding process for reference picture lists construction specified in subclause 8.2.4 or the decoded reference picture marking process specified in subclause 8.2.5 is invoked.

The variables `FrameNum`, `FrameNumWrap`, `PicNum`, `LongTermFrameIdx`, and `LongTermPicNum` are used for the initialisation process for reference picture lists in subclause 8.2.4.2, the modification process for reference picture lists in subclause 8.2.4.3, and for the decoded reference picture marking process in subclause 8.2.5.

To each short-term reference picture the variables `FrameNum` and `FrameNumWrap` are assigned as follows. First, `FrameNum` is set equal to the syntax element `frame_num` that has been decoded in the slice header(s) of the corresponding short-term reference picture. Then the variable `FrameNumWrap` is derived as

$$\begin{aligned} &\text{if(FrameNum > frame_num)} \\ &\quad \text{FrameNumWrap} = \text{FrameNum} - \text{MaxFrameNum} \\ &\text{else} \\ &\quad \text{FrameNumWrap} = \text{FrameNum} \end{aligned} \tag{8-28}$$

where the value of `frame_num` used in Equation 8-28 is the `frame_num` in the slice header(s) for the current picture.

Each long-term reference picture has an associated value of `LongTermFrameIdx` (that was assigned to it as specified in subclause 8.2.5).

To each short-term reference picture a variable `PicNum` is assigned, and to each long-term reference picture a variable `LongTermPicNum` is assigned. The values of these variables depend on the value of `field_pic_flag` and `bottom_field_flag` for the current picture and they are set as follows.

- If `field_pic_flag` is equal to 0, the following applies.

- For each short-term reference frame or complementary reference field pair:

$$\text{PicNum} = \text{FrameNumWrap} \tag{8-29}$$

- For each long-term reference frame or long-term complementary reference field pair:

$$\text{LongTermPicNum} = \text{LongTermFrameIdx} \tag{8-30}$$

NOTE – When decoding a frame the value of `MbaffFrameFlag` has no influence on the derivations in subclauses 8.2.4.2, 8.2.4.3, and 8.2.5.

- Otherwise (`field_pic_flag` is equal to 1), the following applies.

- For each short-term reference field the following applies.

- If the reference field has the same parity as the current field

$$\text{PicNum} = 2 * \text{FrameNumWrap} + 1 \tag{8-31}$$

- Otherwise (the reference field has the opposite parity of the current field),

$$\text{PicNum} = 2 * \text{FrameNumWrap} \tag{8-32}$$

- For each long-term reference field the following applies.

- If the reference field has the same parity as the current field

$$\text{LongTermPicNum} = 2 * \text{LongTermFrameIdx} + 1 \tag{8-33}$$

- Otherwise (the reference field has the opposite parity of the current field),

$$\text{LongTermPicNum} = 2 * \text{LongTermFrameIdx} \tag{8-34}$$

8.2.4.2 Initialisation process for reference picture lists

This initialisation process is invoked when decoding a P, SP, or B slice header.

RefPicList0 and RefPicList1 have initial entries as specified in subclauses 8.2.4.2.1 through 8.2.4.2.5.

When the number of entries in the initial RefPicList0 or RefPicList1 produced as specified in subclauses 8.2.4.2.1 through 8.2.4.2.5 is greater than $\text{num_ref_idx_l0_active_minus1} + 1$ or $\text{num_ref_idx_l1_active_minus1} + 1$, respectively, the extra entries past position $\text{num_ref_idx_l0_active_minus1}$ or $\text{num_ref_idx_l1_active_minus1}$ are discarded from the initial reference picture list.

When the number of entries in the initial RefPicList0 or RefPicList1 produced as specified in subclauses 8.2.4.2.1 through 8.2.4.2.5 is less than $\text{num_ref_idx_l0_active_minus1} + 1$ or $\text{num_ref_idx_l1_active_minus1} + 1$, respectively, the remaining entries in the initial reference picture list are set equal to "no reference picture".

8.2.4.2.1 Initialisation process for the reference picture list for P and SP slices in frames

This initialisation process is invoked when decoding a P or SP slice in a coded frame.

When this process is invoked, there shall be at least one reference frame or complementary reference field pair that is currently marked as "used for short-term reference" or "used for long-term reference".

The reference picture list RefPicList0 is ordered so that short-term reference frames and short-term complementary reference field pairs have lower indices than long-term reference frames and long-term complementary reference field pairs.

The short-term reference frames and complementary reference field pairs are ordered starting with the frame or complementary field pair with the highest PicNum value and proceeding through in descending order to the frame or complementary field pair with the lowest PicNum value.

The long-term reference frames and complementary reference field pairs are ordered starting with the frame or complementary field pair with the lowest LongTermPicNum value and proceeding through in ascending order to the frame or complementary field pair with the highest LongTermPicNum value.

NOTE – A non-paired reference field is not used for inter prediction for decoding a frame, regardless of the value of MbaffFrameFlag.

For example, when three reference frames are marked as "used for short-term reference" with PicNum equal to 300, 302, and 303 and two reference frames are marked as "used for long-term reference" with LongTermPicNum equal to 0 and 3, the initial index order is:

- RefPicList0[0] is set equal to the short-term reference picture with PicNum = 303,
- RefPicList0[1] is set equal to the short-term reference picture with PicNum = 302,
- RefPicList0[2] is set equal to the short-term reference picture with PicNum = 300,
- RefPicList0[3] is set equal to the long-term reference picture with LongTermPicNum = 0, and
- RefPicList0[4] is set equal to the long-term reference picture with LongTermPicNum = 3.

8.2.4.2.2 Initialisation process for the reference picture list for P and SP slices in fields

This initialisation process is invoked when decoding a P or SP slice in a coded field.

Each field included in the reference picture list RefPicList0 has a separate index in the reference picture list RefPicList0.

NOTE – When decoding a field, there are effectively at least twice as many pictures available for referencing as there would be when decoding a frame at the same position in decoding order.

Two ordered lists of reference frames, refFrameList0ShortTerm and refFrameList0LongTerm, are derived as follows. For purposes of the formation of this list of frames, decoded reference frames, complementary reference field pairs, non-paired reference fields and reference frames in which a single field is marked "used for short-term reference" or "used for long-term reference" are all considered reference frames.

- All frames having one or more fields marked "used for short-term reference" are included in the list of short-term reference frames refFrameList0ShortTerm. When the current field is the second field (in decoding order) of a complementary reference field pair and the first field is marked as "used for short-term reference", the first field is included in the list of short-term reference frames refFrameList0ShortTerm. refFrameList0ShortTerm is ordered starting with the reference frame with the highest FrameNumWrap value and proceeding through in descending order to the reference frame with the lowest FrameNumWrap value.
- All frames having one or more fields marked "used for long-term reference" are included in the list of long-term reference frames refFrameList0LongTerm. When the current field is the second field (in decoding order) of a complementary reference field pair and the first field is marked as "used for long-term reference", the first field is included in the list of long-term reference frames refFrameList0LongTerm. refFrameList0LongTerm is ordered

starting with the reference frame with the lowest LongTermFrameIdx value and proceeding through in ascending order to the reference frame with the highest LongTermFrameIdx value.

The process specified in subclause 8.2.4.2.5 is invoked with refFrameList0ShortTerm and refFrameList0LongTerm given as input and the output is assigned to RefPicList0.

8.2.4.2.3 Initialisation process for reference picture lists for B slices in frames

This initialisation process is invoked when decoding a B slice in a coded frame.

For purposes of the formation of the reference picture lists RefPicList0 and RefPicList1 the term reference entry refers in the following to decoded reference frames or complementary reference field pairs.

When this process is invoked, there shall be at least one reference entry that is currently marked as "used for short-term reference" or "used for long-term reference".

For B slices, the order of short-term reference entries in the reference picture lists RefPicList0 and RefPicList1 depends on output order, as given by PicOrderCnt(). When pic_order_cnt_type is equal to 0, reference pictures that are marked as "non-existing" as specified in subclause 8.2.5.2 are not included in either RefPicList0 or RefPicList1.

NOTE 1 – When gaps_in_frame_num_value_allowed_flag is equal to 1, encoders should use reference picture list reordering to ensure proper operation of the decoding process (particularly when pic_order_cnt_type is equal to 0, in which case PicOrderCnt() is not inferred for "non-existing" frames).

The reference picture list RefPicList0 is ordered such that short-term reference entries have lower indices than long-term reference entries. It is ordered as follows.

- Let entryShortTerm be a variable ranging over all reference entries that are currently marked as "used for short-term reference". When some values of entryShortTerm are present having PicOrderCnt(entryShortTerm) less than PicOrderCnt(CurrPic), these values of entryShortTerm are placed at the beginning of refPicList0 in descending order of PicOrderCnt(entryShortTerm). All of the remaining values of entryShortTerm (when present) are then appended to refPicList0 in ascending order of PicOrderCnt(entryShortTerm).
- The long-term reference entries are ordered starting with the long-term reference entry that has the lowest LongTermPicNum value and proceeding through in ascending order to the long-term reference entry that has the highest LongTermPicNum value.

The reference picture list RefPicList1 is ordered so that short-term reference entries have lower indices than long-term reference entries. It is ordered as follows.

- Let entryShortTerm be a variable ranging over all reference entries that are currently marked as "used for short-term reference". When some values of entryShortTerm are present having PicOrderCnt(entryShortTerm) greater than PicOrderCnt(CurrPic), these values of entryShortTerm are placed at the beginning of refPicList1 in ascending order of PicOrderCnt(entryShortTerm). All of the remaining values of entryShortTerm (when present) are then appended to refPicList1 in descending order of PicOrderCnt(entryShortTerm).
- Long-term reference entries are ordered starting with the long-term reference frame or complementary reference field pair that has the lowest LongTermPicNum value and proceeding through in ascending order to the long-term reference entry that has the highest LongTermPicNum value.
- When the reference picture list RefPicList1 has more than one entry and RefPicList1 is identical to the reference picture list RefPicList0, the first two entries RefPicList1[0] and RefPicList1[1] are switched.

NOTE 2 – A non-paired reference field is not used for inter prediction of frames (independent of the value of MbaffFrameFlag).

8.2.4.2.4 Initialisation process for reference picture lists for B slices in fields

This initialisation process is invoked when decoding a B slice in a coded field.

When decoding a field, each field of a stored reference frame is identified as a separate reference picture with a unique index. The order of short-term reference pictures in the reference picture lists RefPicList0 and RefPicList1 depend on output order, as given by PicOrderCnt(). When pic_order_cnt_type is equal to 0, reference pictures that are marked as "non-existing" as specified in subclause 8.2.5.2 are not included in either RefPicList0 or RefPicList1.

NOTE 1 – When gaps_in_frame_num_value_allowed_flag is equal to 1, encoders should use reference picture list reordering to ensure proper operation of the decoding process (particularly when pic_order_cnt_type is equal to 0, in which case PicOrderCnt() is not inferred for "non-existing" frames).

NOTE 2 – When decoding a field, there are effectively at least twice as many pictures available for referencing as there would be when decoding a frame at the same position in decoding order.

Three ordered lists of reference frames, refFrameList0ShortTerm, refFrameList1ShortTerm and refFrameListLongTerm, are derived as follows. For purposes of the formation of these lists of frames the term reference entry refers in the following to decoded reference frames, complementary reference field pairs, or non-paired reference

fields. When `pic_order_cnt_type` is equal to 0, the term reference entry does not refer to frames that are marked as "non-existing" as specified in subclause 8.2.5.2.

- Let `entryShortTerm` be a variable ranging over all reference entries that are currently marked as "used for short-term reference". When some values of `entryShortTerm` are present having `PicOrderCnt(entryShortTerm)` less than or equal to `PicOrderCnt(CurrPic)`, these values of `entryShortTerm` are placed at the beginning of `refFrameList0ShortTerm` in descending order of `PicOrderCnt(entryShortTerm)`. All of the remaining values of `entryShortTerm` (when present) are then appended to `refFrameList0ShortTerm` in ascending order of `PicOrderCnt(entryShortTerm)`.

NOTE 3 – When the current field follows in decoding order a coded field `fldPrev` with which together it forms a complementary reference field pair, `fldPrev` is included into the list `refFrameList0ShortTerm` using `PicOrderCnt(fldPrev)` and the ordering method described in the previous sentence is applied.

- Let `entryShortTerm` be a variable ranging over all reference entries that are currently marked as "used for short-term reference". When some values of `entryShortTerm` are present having `PicOrderCnt(entryShortTerm)` greater than `PicOrderCnt(CurrPic)`, these values of `entryShortTerm` are placed at the beginning of `refFrameList1ShortTerm` in ascending order of `PicOrderCnt(entryShortTerm)`. All of the remaining values of `entryShortTerm` (when present) are then appended to `refFrameList1ShortTerm` in descending order of `PicOrderCnt(entryShortTerm)`.

NOTE 4 – When the current field follows in decoding order a coded field `fldPrev` with which together it forms a complementary reference field pair, `fldPrev` is included into the list `refFrameList1ShortTerm` using `PicOrderCnt(fldPrev)` and the ordering method described in the previous sentence is applied.

- `refFrameListLongTerm` is ordered starting with the reference entry having the lowest `LongTermFrameIdx` value and proceeding through in ascending order to the reference entry having highest `LongTermFrameIdx` value.

NOTE 5 – When the complementary field of the current picture is marked "used for long-term reference" it is included into the list `refFrameListLongTerm`. A reference entry in which only one field is marked as "used for long-term reference" is included into the list `refFrameListLongTerm`.

The process specified in subclause 8.2.4.2.5 is invoked with `refFrameList0ShortTerm` and `refFrameListLongTerm` given as input and the output is assigned to `RefPicList0`.

The process specified in subclause 8.2.4.2.5 is invoked with `refFrameList1ShortTerm` and `refFrameListLongTerm` given as input and the output is assigned to `RefPicList1`.

When the reference picture list `RefPicList1` has more than one entry and `RefPicList1` is identical to the reference picture list `RefPicList0`, the first two entries `RefPicList1[0]` and `RefPicList1[1]` are switched.

8.2.4.2.5 Initialisation process for reference picture lists in fields

Inputs of this process are the reference frame lists `refFrameListXShortTerm` (with `X` may be 0 or 1) and `refFrameListLongTerm`.

The reference picture list `RefPicListX` is a list ordered such that short-term reference fields have lower indices than long-term reference fields. Given the reference frame lists `refFrameListXShortTerm` and `refFrameListLongTerm`, it is derived as follows.

- Short-term reference fields are ordered by selecting reference fields from the ordered list of frames `refFrameListXShortTerm` by alternating between fields of differing parity, starting with a field that has the same parity as the current field (when present). When one field of a reference frame was not decoded or is not marked as "used for short-term reference", the missing field is ignored and instead the next available stored reference field of the chosen parity from the ordered list of frames `refFrameListXShortTerm` is inserted into `RefPicListX`. When there are no more short-term reference fields of the alternate parity in the ordered list of frames `refFrameListXShortTerm`, the next not yet indexed fields of the available parity are inserted into `RefPicListX` in the order in which they occur in the ordered list of frames `refFrameListXShortTerm`.
- Long-term reference fields are ordered by selecting reference fields from the ordered list of frames `refFrameListLongTerm` by alternating between fields of differing parity, starting with a field that has the same parity as the current field (when present). When one field of a reference frame was not decoded or is not marked as "used for long-term reference", the missing field is ignored and instead the next available stored reference field of the chosen parity from the ordered list of frames `refFrameListLongTerm` is inserted into `RefPicListX`. When there are no more long-term reference fields of the alternate parity in the ordered list of frames `refFrameListLongTerm`, the next not yet indexed fields of the available parity are inserted into `RefPicListX` in the order in which they occur in the ordered list of frames `refFrameListLongTerm`.

8.2.4.3 Reordering process for reference picture lists

When `ref_pic_list_reordering_flag_l0` is equal to 1, the following applies.

- Let `refIdxL0` be an index into the reference picture list `RefPicList0`. It is initially set equal to 0.

- The corresponding syntax elements `reordering_of_pic_nums_idc` are processed in the order they occur in the bitstream. For each of these syntax elements, the following applies.
 - If `reordering_of_pic_nums_idc` is equal to 0 or equal to 1, the process specified in subclause 8.2.4.3.1 is invoked with `refIdxL0` as input, and the output is assigned to `refIdxL0`.
 - Otherwise, if `reordering_of_pic_nums_idc` is equal to 2, the process specified in subclause 8.2.4.3.2 is invoked with `refIdxL0` as input, and the output is assigned to `refIdxL0`.
 - Otherwise (`reordering_of_pic_nums_idc` is equal to 3), the reordering process for reference picture list `RefPicList0` is finished.

When `ref_pic_list_reordering_flag_l1` is equal to 1, the following applies.

- Let `refIdxL1` be an index into the reference picture list `RefPicList1`. It is initially set equal to 0.
- The corresponding syntax elements `reordering_of_pic_nums_idc` are processed in the order they occur in the bitstream. For each of these syntax elements, the following applies.
 - If `reordering_of_pic_nums_idc` is equal to 0 or equal to 1, the process specified in subclause 8.2.4.3.1 is invoked with `refIdxL1` as input, and the output is assigned to `refIdxL1`.
 - Otherwise, if `reordering_of_pic_nums_idc` is equal to 2, the process specified in subclause 8.2.4.3.2 is invoked with `refIdxL1` as input, and the output is assigned to `refIdxL1`.
 - Otherwise (`reordering_of_pic_nums_idc` is equal to 3), the reordering process for reference picture list `RefPicList1` is finished.

8.2.4.3.1 Reordering process of reference picture lists for short-term reference pictures

Input to this process is an index `refIdxLX` (with `X` being 0 or 1).

Output of this process is an incremented index `refIdxLX`.

The variable `picNumLXNoWrap` is derived as follows.

- If `reordering_of_pic_nums_idc` is equal to 0

$$\begin{aligned} &\text{if(picNumLXPred - (abs_diff_pic_num_minus1 + 1) < 0)} \\ &\quad \text{picNumLXNoWrap} = \text{picNumLXPred} - (\text{abs_diff_pic_num_minus1} + 1) + \text{MaxPicNum} \\ &\text{else} \\ &\quad \text{picNumLXNoWrap} = \text{picNumLXPred} - (\text{abs_diff_pic_num_minus1} + 1) \end{aligned} \tag{8-35}$$

- Otherwise (`reordering_of_pic_nums_idc` is equal to 1),

$$\begin{aligned} &\text{if(picNumLXPred + (abs_diff_pic_num_minus1 + 1) \geq \text{MaxPicNum})} \\ &\quad \text{picNumLXNoWrap} = \text{picNumLXPred} + (\text{abs_diff_pic_num_minus1} + 1) - \text{MaxPicNum} \\ &\text{else} \\ &\quad \text{picNumLXNoWrap} = \text{picNumLXPred} + (\text{abs_diff_pic_num_minus1} + 1) \end{aligned} \tag{8-36}$$

`picNumLXPred` is the prediction value for the variable `picNumLXNoWrap`. When the process specified in this subclause is invoked the first time for a slice (that is, for the first occurrence of `reordering_of_pic_nums_idc` equal to 0 or 1 in the `ref_pic_list_reordering()` syntax), `picNumL0Pred` and `picNumL1Pred` are initially set equal to `CurrPicNum`. After each assignment of `picNumLXNoWrap`, the value of `picNumLXNoWrap` is assigned to `picNumLXPred`.

The variable `picNumLX` is derived as follows

$$\begin{aligned} &\text{if(picNumLXNoWrap > CurrPicNum)} \\ &\quad \text{picNumLX} = \text{picNumLXNoWrap} - \text{MaxPicNum} \\ &\text{else} \\ &\quad \text{picNumLX} = \text{picNumLXNoWrap} \end{aligned} \tag{8-37}$$

`picNumLX` shall be equal to the `PicNum` of a reference picture that is marked as “used for short-term reference” and shall not be equal to the `PicNum` of a short-term reference picture that is marked as “non-existing”.

The following procedure is conducted to place the picture with short-term picture number `picNumLX` into the index position `refIdxLX`, shift the position of any other remaining pictures to later in the list, and increment the value of `refIdxLX`.

```

for( cIdx = num_ref_idx_lX_active_minus1 + 1; cIdx > refIdxLX; cIdx-- )
    RefPicListX[ cIdx ] = RefPicListX[ cIdx - 1 ]
RefPicListX[ refIdxLX++ ] = short-term reference picture with PicNum equal to picNumLX
nIdx = refIdxLX
for( cIdx = refIdxLX; cIdx <= num_ref_idx_lX_active_minus1 + 1; cIdx++ )
    if( PicNumF( RefPicListX[ cIdx ] ) != picNumLX )
        RefPicListX[ nIdx++ ] = RefPicListX[ cIdx ]

```

(8-38)

where the function PicNumF(RefPicListX[cIdx]) is derived as follows:

- If the picture RefPicListX[cIdx] is marked as "used for short-term reference", PicNumF(RefPicListX[cIdx]) is the PicNum of the picture RefPicListX[cIdx].
- Otherwise (the picture RefPicListX[cIdx] is not marked as "used for short-term reference"), PicNumF(RefPicListX[cIdx]) is equal to MaxPicNum.

NOTE 1 – A value of MaxPicNum can never be equal to picNumLX.

NOTE 2 – Within this pseudo-code procedure, the length of the list RefPicListX is temporarily made one element longer than the length needed for the final list. After the execution of this procedure, only elements 0 through num_ref_idx_lX_active_minus1 of the list need to be retained.

8.2.4.3.2 Reordering process of reference picture lists for long-term reference pictures

Input to this process is an index refIdxLX (with X being 0 or 1).

Output of this process is an incremented index refIdxLX.

The following procedure is conducted to place the picture with long-term picture number long_term_pic_num into the index position refIdxLX, shift the position of any other remaining pictures to later in the list, and increment the value of refIdxLX.

```

for( cIdx = num_ref_idx_lX_active_minus1 + 1; cIdx > refIdxLX; cIdx-- )
    RefPicListX[ cIdx ] = RefPicListX[ cIdx - 1 ]
RefPicListX[ refIdxLX++ ] = long-term reference picture with LongTermPicNum equal to long_term_pic_num
nIdx = refIdxLX
for( cIdx = refIdxLX; cIdx <= num_ref_idx_lX_active_minus1 + 1; cIdx++ )
    if( LongTermPicNumF( RefPicListX[ cIdx ] ) != long_term_pic_num )
        RefPicListX[ nIdx++ ] = RefPicListX[ cIdx ]

```

(8-39)

where the function LongTermPicNumF(RefPicListX[cIdx]) is derived as follows:

- If the picture RefPicListX[cIdx] is marked as "used for long-term reference", LongTermPicNumF(RefPicListX[cIdx]) is the LongTermPicNum of the picture RefPicListX[cIdx].
- Otherwise (the picture RefPicListX[cIdx] is not marked as "used for long-term reference"), LongTermPicNumF(RefPicListX[cIdx]) is equal to 2 * (MaxLongTermFrameIdx + 1).

NOTE 1 – A value of 2 * (MaxLongTermFrameIdx + 1) can never be equal to long_term_pic_num.

NOTE 2 – Within this pseudo-code procedure, the length of the list RefPicListX is temporarily made one element longer than the length needed for the final list. After the execution of this procedure, only elements 0 through num_ref_idx_lX_active_minus1 of the list need to be retained.

8.2.5 Decoded reference picture marking process

This process is invoked for decoded pictures when nal_ref_idc is not equal to 0.

NOTE – The decoding process for gaps in frame_num that is specified in subclause 8.2.5.2 may also be invoked when nal_ref_idc is equal to 0, as specified in clause 8.

A decoded picture with nal_ref_idc not equal to 0, referred to as a reference picture, is marked as "used for short-term reference" or "used for long-term reference". For a decoded reference frame, both of its fields are marked the same as the frame. For a complementary reference field pair, the pair is marked the same as both of its fields. A picture that is marked as "used for short-term reference" is identified by its FrameNum and, when it is a field, by its parity. A picture that is marked as "used for long-term reference" is identified by its LongTermFrameIdx and, when it is a field, by its parity.

Frames or complementary field pairs marked as "used for short-term reference" or as "used for long-term reference" can be used as a reference for inter prediction when decoding a frame until the frame, the complementary field pair, or one of its constituent fields is marked as "unused for reference". A field marked as "used for short-term reference" or as

"used for long-term reference" can be used as a reference for inter prediction when decoding a field until marked as "unused for reference".

A picture can be marked as "unused for reference" by the sliding window reference picture marking process, a first-in, first-out mechanism specified in subclause 8.2.5.3 or by the adaptive memory control reference picture marking process, a customised adaptive marking operation specified in subclause 8.2.5.4.

A short-term reference picture is identified for use in the decoding process by its variables `FrameNum` and `FrameNumWrap` and its picture number `PicNum`, and a long-term reference picture is identified for use in the decoding process by its long-term picture number `LongTermPicNum`. When the current picture is not an IDR picture, subclause 8.2.4.1 is invoked to specify the assignment of the variables `FrameNum`, `FrameNumWrap`, `PicNum` and `LongTermPicNum`.

8.2.5.1 Sequence of operations for decoded reference picture marking process

Decoded reference picture marking proceeds in the following ordered steps.

1. All slices of the current picture are decoded.
2. Depending on whether the current picture is an IDR picture, the following applies.
 - If the current picture is an IDR picture, the following applies.
 - All reference pictures are marked as "unused for reference"
 - Depending on `long_term_reference_flag`, the following applies.
 - If `long_term_reference_flag` is equal to 0, the IDR picture is marked as "used for short-term reference" and `MaxLongTermFrameIdx` is set equal to "no long-term frame indices".
 - Otherwise (`long_term_reference_flag` is equal to 1), the IDR picture is marked as "used for long-term reference", the `LongTermFrameIdx` for the IDR picture is set equal to 0, and `MaxLongTermFrameIdx` is set equal to 0.
 - Otherwise (the current picture is not an IDR picture), the following applies.
 - If `adaptive_ref_pic_marking_mode_flag` is equal to 0, the process specified in subclause 8.2.5.3 is invoked.
 - Otherwise (`adaptive_ref_pic_marking_mode_flag` is equal to 1), the process specified in subclause 8.2.5.4 is invoked.
3. When the current picture is not an IDR picture and it was not marked as "used for long-term reference" by `memory_management_control_operation` equal to 6, it is marked as "used for short-term reference".

After marking the current decoded reference picture, the total number of frames with at least one field marked as "used for reference", plus the number of complementary field pairs with at least one field marked as "used for reference", plus the number of non-paired fields marked as "used for reference" shall not be greater than $\text{Max}(\text{num_ref_frames}, 1)$.

8.2.5.2 Decoding process for gaps in `frame_num`

This process is invoked when `frame_num` is not equal to `PrevRefFrameNum` and is not equal to $(\text{PrevRefFrameNum} + 1) \% \text{MaxFrameNum}$.

NOTE 1 – Although this process is specified as a subclause within subclause 8.2.5 (which defines a process that is invoked only when `nal_ref_idc` is not equal to 0), this process may also be invoked when `nal_ref_idc` is equal to 0 (as specified in clause 8). The reasons for the location of this subclause within the structure of this Recommendation | International Standard are historical.

NOTE 2 – This process can only be invoked for a conforming bitstream when `gaps_in_frame_num_value_allowed_flag` is equal to 1. When `gaps_in_frame_num_value_allowed_flag` is equal to 0 and `frame_num` is not equal to `PrevRefFrameNum` and is not equal to $(\text{PrevRefFrameNum} + 1) \% \text{MaxFrameNum}$, the decoding process should infer an unintentional loss of pictures.

When this process is invoked, a set of values of `frame_num` pertaining to "non-existing" pictures is derived as all values taken on by `UnusedShortTermFrameNum` in Equation 7-21 except the value of `frame_num` for the current picture.

The decoding process generates and marks a frame for each of the values of `frame_num` pertaining to "non-existing" pictures, in the order in which the values of `UnusedShortTermFrameNum` are generated by Equation 7-21, using the "sliding window" picture marking process as specified in subclause 8.2.5.3. The generated frames are also marked as "non-existing" and "used for short-term reference". The sample values of the generated frames may be set to any value. The bitstream shall not contain data that results in a reference to these generated frames which are marked as "non-existing" in the inter prediction process, a reference to these frames in the reordering commands for reference picture lists for short-term reference pictures (subclause 8.2.4.3.1), or a reference to these frames in the assignment process of a `LongTermFrameIdx` to a short-term reference picture (subclause 8.2.5.4.3).

When `pic_order_cnt_type` is not equal to 0, `TopFieldOrderCnt` and `BottomFieldOrderCnt` are derived for each of the "non-existing" frames by invoking the decoding process for picture order count in subclause 8.2.1. When invoking the process in subclause 8.2.1 for a particular "non-existing" frame, the current picture is considered to be a picture considered having `frame_num` inferred to be equal to `UnusedShortTermFrameNum`, `nal_ref_idc` inferred to be not equal to 0, `nal_unit_type` inferred to be not equal to 5, `field_pic_flag` inferred to be equal to 0, `adaptive_ref_pic_marking_mode_flag` inferred to be equal to 0, `delta_pic_order_cnt[0]` (if needed) inferred to be equal to 0, and `delta_pic_order_cnt[1]` (if needed) inferred to be equal to 0.

NOTE 3 – The decoding process should infer an unintentional picture loss when any of these values of `frame_num` pertaining to "non-existing" pictures is referred to in the inter prediction process, is referred to in the reordering commands for reference picture lists for short-term reference pictures (subclause 8.2.4.3.1), or is referred to in the assignment process of a `LongTermFrameIdx` to a short-term reference picture (subclause 8.2.5.4.3). The decoding process should not infer an unintentional picture loss when a memory management control operation not equal to 3 is applied to a frame marked as "non-existing".

8.2.5.3 Sliding window decoded reference picture marking process

This process is invoked when `adaptive_ref_pic_marking_mode_flag` is equal to 0.

Depending on the properties of the current picture as specified below, the following applies.

- If the current picture is a coded field that is the second field in decoding order of a complementary reference field pair, and the first field has been marked as "used for short-term reference", the current picture is also marked as "used for short-term reference".
- Otherwise, the following applies.
 - Let `numShortTerm` be the total number of reference frames, complementary reference field pairs and non-paired reference fields for which at least one field is marked as "used for short-term reference". Let `numLongTerm` be the total number of reference frames, complementary reference field pairs and non-paired reference fields for which at least one field is marked as "used for long-term reference".
 - When `numShortTerm + numLongTerm` is equal to `Max(num_ref_frames, 1)`, the condition that `numShortTerm` is greater than 0 shall be fulfilled, and the short-term reference frame, complementary reference field pair or non-paired reference field that has the smallest value of `FrameNumWrap` is marked as "unused for reference". When it is a frame or a complementary field pair, both of its fields are also marked as "unused for reference".

8.2.5.4 Adaptive memory control decoded reference picture marking process

This process is invoked when `adaptive_ref_pic_marking_mode_flag` is equal to 1.

The `memory_management_control_operation` commands with values of 1 to 6 are processed in the order they occur in the bitstream after the current picture has been decoded. For each of these `memory_management_control_operation` commands, one of the processes specified in subclauses 8.2.5.4.1 to 8.2.5.4.5 is invoked depending on the value of `memory_management_control_operation`. The `memory_management_control_operation` command with value of 0 specifies the end of `memory_management_control_operation` commands.

Memory management control operations are applied to pictures as follows.

- If `field_pic_flag` is equal to 0, `memory_management_control_operation` commands are applied to the frames or complementary reference field pairs specified.
- Otherwise (`field_pic_flag` is equal to 1), `memory_management_control_operation` commands are applied to the individual reference fields specified.

8.2.5.4.1 Marking process of a short-term reference picture as "unused for reference"

This process is invoked when `memory_management_control_operation` is equal to 1.

Let `picNumX` be specified by

$$\text{picNumX} = \text{CurrPicNum} - (\text{difference_of_pic_nums_minus1} + 1). \quad (8-40)$$

Depending on `field_pic_flag` the value of `picNumX` is used to mark a short-term reference picture as "unused for reference" as follows.

- If `field_pic_flag` is equal to 0, the short-term reference frame or short-term complementary reference field pair specified by `picNumX` and both of its fields are marked as "unused for reference".
- Otherwise (`field_pic_flag` is equal to 1), the short-term reference field specified by `picNumX` is marked as "unused for reference". When that reference field is part of a reference frame or a complementary reference field pair, the

frame or complementary field pair is also marked as "unused for reference", but the marking of the other field is not changed.

8.2.5.4.2 Marking process of a long-term reference picture as “unused for reference”

This process is invoked when memory_management_control_operation is equal to 2.

Depending on field_pic_flag the value of LongTermPicNum is used to mark a long-term reference picture as “unused for reference” as follows.

- If field_pic_flag is equal to 0, the long-term reference frame or long-term complementary reference field pair having LongTermPicNum equal to long_term_pic_num and both of its fields are marked as “unused for reference”.
- Otherwise (field_pic_flag is equal to 1), the long-term reference field specified by LongTermPicNum equal to long_term_pic_num is marked as “unused for reference”. When that reference field is part of a reference frame or a complementary reference field pair, the frame or complementary field pair is also marked as "unused for reference", but the marking of the other field is not changed.

8.2.5.4.3 Assignment process of a LongTermFrameIdx to a short-term reference picture

This process is invoked when memory_management_control_operation is equal to 3.

Given the syntax element difference_of_pic_nums_minus1, the variable picNumX is obtained as specified in subclause 8.2.5.4.1. picNumX shall refer to a frame or complementary reference field pair or non-paired reference field marked as "used for short-term reference" and not marked as "non-existing".

When LongTermFrameIdx equal to long_term_frame_idx is already assigned to a long-term reference frame or a long-term complementary reference field pair, that frame or complementary field pair and both of its fields are marked as "unused for reference". When LongTermFrameIdx is already assigned to a non-paired reference field, and the field is not the complementary field of the picture specified by picNumX, that field is marked as “unused for reference”.

Depending on field_pic_flag the value of LongTermFrameIdx is used to mark a picture from "used for short-term reference" to "used for long-term reference" as follows.

- If field_pic_flag is equal to 0, the marking of the short-term reference frame or short-term complementary reference field pair specified by picNumX and both of its fields are changed from "used for short-term reference" to "used for long-term reference" and assigned LongTermFrameIdx equal to long_term_frame_idx.
- Otherwise (field_pic_flag is equal to 1), the marking of the short-term reference field specified by picNumX is changed from "used for short-term reference" to "used for long-term reference" and assigned LongTermFrameIdx equal to long_term_frame_idx. When the field is part of a reference frame or a complementary reference field pair, and the other field of the same reference frame or complementary reference field pair is also marked as "used for long-term reference", the reference frame or complementary reference field pair is also marked as "used for long-term reference" and assigned LongTermFrameIdx equal to long_term_frame_idx.

8.2.5.4.4 Decoding process for MaxLongTermFrameIdx

This process is invoked when memory_management_control_operation is equal to 4.

All pictures for which LongTermFrameIdx is greater than max_long_term_frame_idx_plus1 – 1 and that are marked as "used for long-term reference" are marked as “unused for reference”.

The variable MaxLongTermFrameIdx is derived as follows.

- If max_long_term_frame_idx_plus1 is equal to 0, MaxLongTermFrameIdx is set equal to “no long-term frame indices”.
- Otherwise (max_long_term_frame_idx_plus1 is greater than 0), MaxLongTermFrameIdx is set equal to max_long_term_frame_idx_plus1 – 1.

NOTE – The memory_management_control_operation command equal to 4 can be used to mark long-term reference pictures as “unused for reference”. The frequency of transmitting max_long_term_frame_idx_plus1 is not specified by this Recommendation | International Standard. However, the encoder should send a memory_management_control_operation command equal to 4 upon receiving an error message, such as an intra refresh request message.

8.2.5.4.4.1 Marking process of all reference pictures as “unused for reference” and setting MaxLongTermFrameIdx to “no long-term frame indices”

This process is invoked when memory_management_control_operation is equal to 5.

All reference pictures are marked as “unused for reference” and the variable MaxLongTermFrameIdx is set equal to “no long-term frame indices”.

8.2.5.4.5 Process for assigning a long-term frame index to the current picture

This process is invoked when memory_management_control_operation is equal to 6.

When a variable LongTermFrameIdx equal to long_term_frame_idx is already assigned to a long-term reference frame or a long-term complementary reference field pair, that frame or complementary field pair and both of its fields are marked as "unused for reference". When LongTermFrameIdx is already assigned to a non-paired reference field, and the field is not the complementary field of the current picture, that field is marked as “unused for reference”.

The current picture is marked as "used for long-term reference" and assigned LongTermFrameIdx equal to long_term_frame_idx.

When field_pic_flag is equal to 0, both its fields are also marked as "used for long-term reference" and assigned LongTermFrameIdx equal to long_term_frame_idx.

When field_pic_flag is equal to 1 and the current picture is the second field (in decoding order) of a complementary reference field pair, and the first field of the complementary reference field pair is also currently marked as "used for long-term reference", the complementary reference field pair is also marked as "used for long-term reference" and assigned LongTermFrameIdx equal to long_term_frame_idx.

After marking the current decoded reference picture, the total number of frames with at least one field marked as “used for reference”, plus the number of complementary field pairs with at least one field marked as “used for reference”, plus the number of non-paired fields marked as “used for reference” shall not be greater than Max(num_ref_frames, 1).

NOTE – Under some circumstances, the above statement may impose a constraint on the order in which a memory_management_control_operation syntax element equal to 6 can appear in the decoded reference picture marking syntax relative to a memory_management_control_operation syntax element equal to 1, 2, or 4.

8.3 Intra prediction process

This process is invoked for I and SI macroblock types.

Inputs to this process are constructed samples prior to the deblocking filter process and, for Intra_NxN prediction modes (where NxN is equal to 4x4 or 8x8), the values of IntraNxNPredMode from neighbouring macroblocks.

Outputs of this process are specified as follows.

- If the macroblock prediction mode is Intra_4x4 or Intra_8x8, the outputs are constructed luma samples prior to the deblocking filter process and (when chroma_format_idc is not equal to 0) chroma prediction samples of the macroblock pred_C, where C is equal to Cb and Cr.
- Otherwise, if mb_type is not equal to I_PCM, the outputs are luma prediction samples of the macroblock pred_L and (when chroma_format_idc is not equal to 0) chroma prediction samples of the macroblock pred_C, where C is equal to Cb and Cr.
- Otherwise (mb_type is equal to I_PCM), the outputs are constructed luma and (when chroma_format_idc is not equal to 0) chroma samples prior to the deblocking filter process.

The variable MvCnt is set equal to 0.

Depending on the value of mb_type the following applies.

- If mb_type is equal to I_PCM, the process specified in subclause 8.3.5 is invoked.
- Otherwise (mb_type is not equal to I_PCM), the following applies.
 - The decoding processes for Intra prediction modes are described for the luma component as follows.
 - If the macroblock prediction mode is equal to Intra_4x4, the specification in subclause 8.3.1 applies.
 - Otherwise, if the macroblock prediction mode is equal to Intra_8x8, the specification in subclause 8.3.2 applies.
 - Otherwise (the macroblock prediction mode is equal to Intra_16x16), the specification in subclause 8.3.3 applies.
 - The decoding processes for Intra prediction modes for the chroma components are described in subclause 8.3.4. This process is only invoked when chroma_format_idc is not equal to 0 (monochrome).

Samples used in the Intra prediction process are the sample values prior to alteration by any deblocking filter operation.

8.3.1 Intra_4x4 prediction process for luma samples

This process is invoked when the macroblock prediction mode is equal to Intra_4x4.

Inputs to this process are the values of Intra4x4PredMode (if available) or Intra8x8PredMode (if available) from neighbouring macroblocks or macroblock pairs.

The luma component of a macroblock consists of 16 blocks of 4x4 luma samples. These blocks are inverse scanned using the 4x4 luma block inverse scanning process as specified in subclause 6.4.3.

For all 4x4 luma blocks of the luma component of a macroblock with luma4x4BlkIdx = 0..15, the derivation process for the Intra4x4PredMode as specified in subclause 8.3.1.1 is invoked with luma4x4BlkIdx as well as Intra4x4PredMode and Intra8x8PredMode that are previously (in decoding order) derived for adjacent macroblocks as the input and the variable Intra4x4PredMode[luma4x4BlkIdx] as the output.

For each luma block of 4x4 samples indexed using luma4x4BlkIdx = 0..15,

1. The Intra_4x4 sample prediction process in subclause 8.3.1.2 is invoked with luma4x4BlkIdx and constructed samples prior (in decoding order) to the deblocking filter process from adjacent luma blocks as the input and the output are the Intra_4x4 luma prediction samples $\text{pred}_{4x4L}[x, y]$ with $x, y = 0..3$.
2. The position of the upper-left sample of a 4x4 luma block with index luma4x4BlkIdx inside the current macroblock is derived by invoking the inverse 4x4 luma block scanning process in subclause 6.4.3 with luma4x4BlkIdx as the input and the output being assigned to (xO, yO) and $x, y = 0..3$.

$$\text{pred}_L[xO + x, yO + y] = \text{pred}_{4x4L}[x, y] \quad (8-41)$$

3. The transform coefficient decoding process and picture construction process prior to deblocking filter process in subclause 8.5 is invoked with pred_L and luma4x4BlkIdx as the input and the constructed samples for the current 4x4 luma block S'_L as the output.

8.3.1.1 Derivation process for the Intra4x4PredMode

Inputs to this process are the index of the 4x4 luma block luma4x4BlkIdx and variable arrays Intra4x4PredMode (if available) and Intra8x8PredMode (if available) that are previously (in decoding order) derived for adjacent macroblocks.

Output of this process is the variable Intra4x4PredMode[luma4x4BlkIdx].

Table 8-2 specifies the values for Intra4x4PredMode[luma4x4BlkIdx] and the associated names.

Table 8-2 – Specification of Intra4x4PredMode[luma4x4BlkIdx] and associated names

Intra4x4PredMode[luma4x4BlkIdx]	Name of Intra4x4PredMode[luma4x4BlkIdx]
0	Intra_4x4_Vertical (prediction mode)
1	Intra_4x4_Horizontal (prediction mode)
2	Intra_4x4_DC (prediction mode)
3	Intra_4x4_Diagonal_Down_Left (prediction mode)
4	Intra_4x4_Diagonal_Down_Right (prediction mode)
5	Intra_4x4_Vertical_Right (prediction mode)
6	Intra_4x4_Horizontal_Down (prediction mode)
7	Intra_4x4_Vertical_Left (prediction mode)
8	Intra_4x4_Horizontal_Up (prediction mode)

Intra4x4PredMode[luma4x4BlkIdx] labelled 0, 1, 3, 4, 5, 6, 7, and 8 represent directions of predictions as illustrated in Figure 8-1.

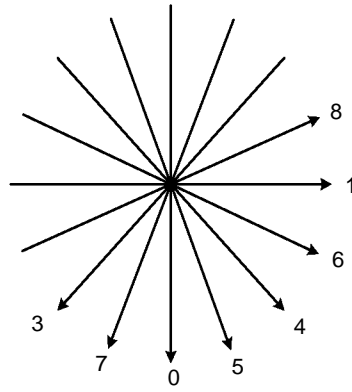


Figure 8-1 – Intra_4x4 prediction mode directions (informative)

$\text{Intra4x4PredMode}[\text{luma4x4BlkIdx}]$ is derived as follows.

- The process specified in subclause 6.4.8.3 is invoked with luma4x4BlkIdx given as input and the output is assigned to mbAddrA , luma4x4BlkIdxA , mbAddrB , and luma4x4BlkIdxB .
- The variable $\text{dcPredModePredictedFlag}$ is derived as follows.
 - If any of the following conditions are true, $\text{dcPredModePredictedFlag}$ is set equal to 1
 - the macroblock with address mbAddrA is not available
 - the macroblock with address mbAddrB is not available
 - the macroblock with address mbAddrA is available and coded in Inter prediction mode and $\text{constrained_intra_pred_flag}$ is equal to 1
 - the macroblock with address mbAddrB is available and coded in Inter prediction mode and $\text{constrained_intra_pred_flag}$ is equal to 1
 - Otherwise, $\text{dcPredModePredictedFlag}$ is set equal to 0.
- For N being either replaced by A or B, the variables intraMxMPredModeN are derived as follows.
 - If $\text{dcPredModePredictedFlag}$ is equal to 1 or the macroblock with address mbAddrN is not coded in Intra_4x4 or Intra_8x8 macroblock prediction mode, intraMxMPredModeN is set equal to 2 (Intra_4x4_DC prediction mode).
 - Otherwise ($\text{dcPredModePredictedFlag}$ is equal to 0 and (the macroblock with address mbAddrN is coded in Intra_4x4 macroblock prediction mode or the macroblock with address mbAddrN is coded in Intra_8x8 macroblock prediction mode)), the following applies.
 - If the macroblock with address mbAddrN is coded in Intra_4x4 macroblock mode, intraMxMPredModeN is set equal to $\text{Intra4x4PredMode}[\text{luma4x4BlkIdxN}]$, where Intra4x4PredMode is the variable array assigned to the macroblock mbAddrN .
 - Otherwise (the macroblock with address mbAddrN is coded in Intra_8x8 macroblock mode), intraMxMPredModeN is set equal to $\text{Intra8x8PredMode}[\text{luma4x4BlkIdxN} \gg 2]$, where Intra8x8PredMode is the variable array assigned to the macroblock mbAddrN .
- $\text{Intra4x4PredMode}[\text{luma4x4BlkIdx}]$ is derived by applying the following procedure.

```

predIntra4x4PredMode = Min( intraMxMPredModeA, intraMxMPredModeB )
if( prev_intra4x4_pred_mode_flag[ luma4x4BlkIdx ] )
    Intra4x4PredMode[ luma4x4BlkIdx ] = predIntra4x4PredMode
else
    if( rem_intra4x4_pred_mode[ luma4x4BlkIdx ] < predIntra4x4PredMode )
        Intra4x4PredMode[ luma4x4BlkIdx ] = rem_intra4x4_pred_mode[ luma4x4BlkIdx ]
    else
        Intra4x4PredMode[ luma4x4BlkIdx ] = rem_intra4x4_pred_mode[ luma4x4BlkIdx ] + 1

```

(8-42)

8.3.1.2 Intra_4x4 sample prediction

This process is invoked for each 4x4 luma block of a macroblock with prediction mode equal to Intra_4x4 followed by the transform decoding process and picture construction process prior to deblocking for each 4x4 luma block.

Input to this process is the index of a 4x4 luma block luma4x4BlkIdx.

Output of this process are the prediction samples $\text{pred4x4}_L[x, y]$, with $x, y = 0..3$ for the 4x4 luma block with index luma4x4BlkIdx.

The position of the upper-left sample of a 4x4 luma block with index luma4x4BlkIdx inside the current macroblock is derived by invoking the inverse 4x4 luma block scanning process in subclause 6.4.3 with luma4x4BlkIdx as the input and the output being assigned to (xO, yO).

The 13 neighbouring samples $p[x, y]$ that are constructed luma samples prior to the deblocking filter process, with $x = -1, y = -1..3$ and $x = 0..7, y = -1$, are derived as follows.

- The luma location (xN, yN) is specified by

$$xN = xO + x \quad (8-43)$$

$$yN = yO + y \quad (8-44)$$

- The derivation process for neighbouring locations in subclause 6.4.9 is invoked for luma locations with (xN, yN) as input and mbAddrN and (xW, yW) as output.
- Each sample $p[x, y]$ with $x = -1, y = -1..3$ and $x = 0..7, y = -1$ is derived as follows.
 - If any of the following conditions is true, the sample $p[x, y]$ is marked as “not available for Intra_4x4 prediction”
 - mbAddrN is not available,
 - the macroblock mbAddrN is coded in Inter prediction mode and constrained_intra_pred_flag is equal to 1.
 - the macroblock mbAddrN has mb_type equal to SI and constrained_intra_pred_flag is equal to 1 and the current macroblock does not have mb_type equal to SI.
 - x is greater than 3 and luma4x4BlkIdx is equal to 3 or 11
 - Otherwise, the sample $p[x, y]$ is marked as “available for Intra_4x4 prediction” and the luma sample at luma location (xW, yW) inside the macroblock mbAddrN is assigned to $p[x, y]$.

When samples $p[x, -1]$, with $x = 4..7$ are marked as “not available for Intra_4x4 prediction,” and the sample $p[3, -1]$ is marked as “available for Intra_4x4 prediction,” the sample value of $p[3, -1]$ is substituted for sample values $p[x, -1]$, with $x = 4..7$ and samples $p[x, -1]$, with $x = 4..7$ are marked as “available for Intra_4x4 prediction”.

NOTE – Each block is assumed to be constructed into a picture array prior to decoding of the next block.

Depending on Intra4x4PredMode[luma4x4BlkIdx], one of the Intra_4x4 prediction modes specified in subclauses 8.3.1.2.1 to 8.3.1.2.9 is invoked.

8.3.1.2.1 Specification of Intra_4x4_Veritical prediction mode

This Intra_4x4 prediction mode is invoked when Intra4x4PredMode[luma4x4BlkIdx] is equal to 0.

This mode shall be used only when the samples $p[x, -1]$ with $x = 0..3$ are marked as “available for Intra_4x4 prediction”.

The values of the prediction samples $\text{pred4x4}_L[x, y]$, with $x, y = 0..3$ are derived by

$$\text{pred4x4}_L[x, y] = p[x, -1], \text{ with } x, y = 0..3 \quad (8-45)$$

8.3.1.2.2 Specification of Intra_4x4_Horizontal prediction mode

This Intra_4x4 prediction mode is invoked when Intra4x4PredMode[luma4x4BlkIdx] is equal to 1.

This mode shall be used only when the samples $p[-1, y]$, with $y = 0..3$ are marked as “available for Intra_4x4 prediction”.

The values of the prediction samples $\text{pred4x4}_L[x, y]$, with $x, y = 0..3$ are derived by

$$\text{pred4x4}_L[x, y] = p[-1, y], \text{ with } x, y = 0..3 \quad (8-46)$$

8.3.1.2.3 Specification of Intra_4x4_DC prediction mode

This Intra_4x4 prediction mode is invoked when $\text{Intra4x4PredMode}[\text{luma4x4BlkIdx}]$ is equal to 2.

The values of the prediction samples $\text{pred4x4}_L[x, y]$, with $x, y = 0..3$ are derived as follows.

- If all samples $p[x, -1]$, with $x = 0..3$ and $p[-1, y]$, with $y = 0..3$ are marked as “available for Intra_4x4 prediction”, the values of the prediction samples $\text{pred4x4}_L[x, y]$, with $x, y = 0..3$ are derived by

$$\text{pred4x4}_L[x, y] = (p[0, -1] + p[1, -1] + p[2, -1] + p[3, -1] + p[-1, 0] + p[-1, 1] + p[-1, 2] + p[-1, 3] + 4) \gg 3 \quad (8-47)$$

- Otherwise, if any samples $p[x, -1]$, with $x = 0..3$ are marked as “not available for Intra_4x4 prediction” and all samples $p[-1, y]$, with $y = 0..3$ are marked as “available for Intra_4x4 prediction”, the values of the prediction samples $\text{pred4x4}_L[x, y]$, with $x, y = 0..3$ are derived by

$$\text{pred4x4}_L[x, y] = (p[-1, 0] + p[-1, 1] + p[-1, 2] + p[-1, 3] + 2) \gg 2 \quad (8-48)$$

- Otherwise, if any samples $p[-1, y]$, with $y = 0..3$ are marked as “not available for Intra_4x4 prediction” and all samples $p[x, -1]$, with $x = 0..3$ are marked as “available for Intra_4x4 prediction”, the values of the prediction samples $\text{pred4x4}_L[x, y]$, with $x, y = 0..3$ are derived by

$$\text{pred4x4}_L[x, y] = (p[0, -1] + p[1, -1] + p[2, -1] + p[3, -1] + 2) \gg 2 \quad (8-49)$$

- Otherwise (some samples $p[x, -1]$, with $x = 0..3$ and some samples $p[-1, y]$, with $y = 0..3$ are marked as “not available for Intra_4x4 prediction”), the values of the prediction samples $\text{pred4x4}_L[x, y]$, with $x, y = 0..3$ are derived by

$$\text{pred4x4}_L[x, y] = (1 \ll (\text{BitDepth}_Y - 1)) \quad (8-50)$$

NOTE – A 4x4 luma block can always be predicted using this mode.

8.3.1.2.4 Specification of Intra_4x4_Diagonal_Down_Left prediction mode

This Intra_4x4 prediction mode is invoked when $\text{Intra4x4PredMode}[\text{luma4x4BlkIdx}]$ is equal to 3.

This mode shall be used only when the samples $p[x, -1]$ with $x = 0..7$ are marked as “available for Intra_4x4 prediction”.

The values of the prediction samples $\text{pred4x4}_L[x, y]$, with $x, y = 0..3$ are derived as follows.

- If x is equal to 3 and y is equal to 3,

$$\text{pred4x4}_L[x, y] = (p[6, -1] + 3 * p[7, -1] + 2) \gg 2 \quad (8-51)$$

- Otherwise (x is not equal to 3 or y is not equal to 3),

$$\text{pred4x4}_L[x, y] = (p[x + y, -1] + 2 * p[x + y + 1, -1] + p[x + y + 2, -1] + 2) \gg 2 \quad (8-52)$$

8.3.1.2.5 Specification of Intra_4x4_Diagonal_Down_Right prediction mode

This Intra_4x4 prediction mode is invoked when $\text{Intra4x4PredMode}[\text{luma4x4BlkIdx}]$ is equal to 4.

This mode shall be used only when the samples $p[x, -1]$ with $x = 0..3$ and $p[-1, y]$ with $y = -1..3$ are marked as “available for Intra_4x4 prediction”.

The values of the prediction samples $\text{pred4x4}_L[x, y]$, with $x, y = 0..3$ are derived as follows.

- If x is greater than y ,

$$\text{pred4x4}_L[x, y] = (p[x - y - 2, -1] + 2 * p[x - y - 1, -1] + p[x - y, -1] + 2) >> 2 \quad (8-53)$$

- Otherwise if x is less than y,

$$\text{pred4x4}_L[x, y] = (p[-1, y - x - 2] + 2 * p[-1, y - x - 1] + p[-1, y - x] + 2) >> 2 \quad (8-54)$$

- Otherwise (x is equal to y),

$$\text{pred4x4}_L[x, y] = (p[0, -1] + 2 * p[-1, -1] + p[-1, 0] + 2) >> 2 \quad (8-55)$$

8.3.1.2.6 Specification of Intra_4x4_Vertical_Right prediction mode

This Intra_4x4 prediction mode is invoked when Intra4x4PredMode[luma4x4BlkIdx] is equal to 5.

This mode shall be used only when the samples $p[x, -1]$ with $x = 0..3$ and $p[-1, y]$ with $y = -1..3$ are marked as “available for Intra_4x4 prediction”.

Let the variable zVR be set equal to $2 * x - y$.

The values of the prediction samples $\text{pred4x4}_L[x, y]$, with $x, y = 0..3$ are derived as follows.

- If zVR is equal to 0, 2, 4, or 6,

$$\text{pred4x4}_L[x, y] = (p[x - (y >> 1) - 1, -1] + p[x - (y >> 1), -1] + 1) >> 1 \quad (8-56)$$

- Otherwise, if zVR is equal to 1, 3, or 5,

$$\text{pred4x4}_L[x, y] = (p[x - (y >> 1) - 2, -1] + 2 * p[x - (y >> 1) - 1, -1] + p[x - (y >> 1), -1] + 2) >> 2 \quad (8-57)$$

- Otherwise, if zVR is equal to -1,

$$\text{pred4x4}_L[x, y] = (p[-1, 0] + 2 * p[-1, -1] + p[0, -1] + 2) >> 2 \quad (8-58)$$

- Otherwise (zVR is equal to -2 or -3),

$$\text{pred4x4}_L[x, y] = (p[-1, y - 1] + 2 * p[-1, y - 2] + p[-1, y - 3] + 2) >> 2 \quad (8-59)$$

8.3.1.2.7 Specification of Intra_4x4_Horizontal_Down prediction mode

This Intra_4x4 prediction mode is invoked when Intra4x4PredMode[luma4x4BlkIdx] is equal to 6.

This mode shall be used only when the samples $p[x, -1]$ with $x = 0..3$ and $p[-1, y]$ with $y = -1..3$ are marked as “available for Intra_4x4 prediction”.

Let the variable zHD be set equal to $2 * y - x$.

The values of the prediction samples $\text{pred4x4}_L[x, y]$, with $x, y = 0..3$ are derived as follows.

- If zHD is equal to 0, 2, 4, or 6,

$$\text{pred4x4}_L[x, y] = (p[-1, y - (x >> 1) - 1] + p[-1, y - (x >> 1)] + 1) >> 1 \quad (8-60)$$

- Otherwise, if zHD is equal to 1, 3, or 5,

$$\text{pred4x4}_L[x, y] = (p[-1, y - (x >> 1) - 2] + 2 * p[-1, y - (x >> 1) - 1] + p[-1, y - (x >> 1)] + 2) >> 2 \quad (8-61)$$

- Otherwise, if zHD is equal to -1,

$$\text{pred4x4}_L[x, y] = (p[-1, 0] + 2 * p[-1, -1] + p[0, -1] + 2) >> 2 \quad (8-62)$$

- Otherwise (zHD is equal to -2 or -3),

$$\text{pred4x4}_L[x, y] = (p[x - 1, -1] + 2 * p[x - 2, -1] + p[x - 3, -1] + 2) >> 2 \quad (8-63)$$

8.3.1.2.8 Specification of Intra_4x4_Vertical_Left prediction mode

This Intra_4x4 prediction mode is invoked when Intra4x4PredMode[luma4x4BlkIdx] is equal to 7.

This mode shall be used only when the samples $p[x, -1]$ with $x = 0..7$ are marked as “available for Intra_4x4 prediction”.

The values of the prediction samples $\text{pred4x4}_L[x, y]$, with $x, y = 0..3$ are derived as follows.

- If y is equal to 0 or 2,

$$\text{pred4x4}_L[x, y] = (p[x + (y >> 1), -1] + p[x + (y >> 1) + 1, -1] + 1) >> 1 \quad (8-64)$$

- Otherwise (y is equal to 1 or 3),

$$\text{pred4x4}_L[x, y] = (p[x + (y >> 1), -1] + 2 * p[x + (y >> 1) + 1, -1] + p[x + (y >> 1) + 2, -1] + 2) >> 2 \quad (8-65)$$

8.3.1.2.9 Specification of Intra_4x4_Horizontal_Up prediction mode

This Intra_4x4 prediction mode is invoked when Intra4x4PredMode[luma4x4BlkIdx] is equal to 8.

This mode shall be used only when the samples $p[-1, y]$ with $y = 0..3$ are marked as “available for Intra_4x4 prediction”.

Let the variable zHU be set equal to $x + 2 * y$.

The values of the prediction samples $\text{pred4x4}_L[x, y]$, with $x, y = 0..3$ are derived as follows:

- If zHU is equal to 0, 2, or 4

$$\text{pred4x4}_L[x, y] = (p[-1, y + (x >> 1)] + p[-1, y + (x >> 1) + 1] + 1) >> 1 \quad (8-66)$$

- Otherwise, if zHU is equal to 1 or 3

$$\text{pred4x4}_L[x, y] = (p[-1, y + (x >> 1)] + 2 * p[-1, y + (x >> 1) + 1] + p[-1, y + (x >> 1) + 2] + 2) >> 2 \quad (8-67)$$

- Otherwise, if zHU is equal to 5,

$$\text{pred4x4}_L[x, y] = (p[-1, 2] + 3 * p[-1, 3] + 2) >> 2 \quad (8-68)$$

- Otherwise (zHU is greater than 5),

$$\text{pred4x4}_L[x, y] = p[-1, 3] \quad (8-69)$$

8.3.2 Intra_8x8 prediction process for luma samples

This process is invoked when the macroblock prediction mode is equal to Intra_8x8.

Inputs to this process are the values of Intra4x4PredMode (if available) or Intra8x8PredMode (if available) from the neighbouring macroblocks or macroblock pairs.

Outputs of this process are 8x8 luma sample arrays as part of the 16x16 luma array of prediction samples of the macroblock pred_L .

The luma component of a macroblock consists of 4 blocks of 8x8 luma samples. These blocks are inverse scanned using the inverse 8x8 luma block scanning process as specified in subclause 6.4.4.

For all 8x8 luma blocks of the luma component of a macroblock with $\text{luma8x8BlkIdx} = 0..3$, the derivation process for Intra8x8PredMode as specified in subclause 8.3.2.1 is invoked with luma8x8BlkIdx as well as Intra4x4PredMode and

Intra8x8PredMode that are previously (in decoding order) derived for adjacent macroblocks as the input and the variable Intra8x8PredMode[luma8x8BlkIdx] as the output.

For each luma block of 8x8 samples indexed using luma8x8BlkIdx = 0..3, the following applies.

- The Intra_8x8 sample prediction process in subclause 8.3.2.2 is invoked with luma8x8BlkIdx and constructed samples prior (in decoding order) to the deblocking filter process from adjacent luma blocks as the input and the output are the Intra_8x8 luma prediction samples pred8x8L[x, y] with x, y = 0..7.
- The position of the upper-left sample of an 8x8 luma block with index luma8x8BlkIdx inside the current macroblock is derived by invoking the inverse 8x8 luma block scanning process in subclause 6.4.4 with luma8x8BlkIdx as the input and the output being assigned to (xO, yO) and x, y = 0..7.

$$\text{pred}_L[xO + x, yO + y] = \text{pred8x8}_L[x, y] \quad (8-70)$$

- The transform coefficient decoding process and picture construction process prior to deblocking filter process in subclause 8.5 is invoked with pred_L and luma8x8BlkIdx as the input and the constructed samples for the current 8x8 luma block S'_L as the output.

8.3.2.1 Derivation process for Intra8x8PredMode

Inputs to this process are the index of the 8x8 luma block luma8x8BlkIdx and variable arrays Intra4x4PredMode (if available) and Intra8x8PredMode (if available) that are previously (in decoding order) derived for adjacent macroblocks.

Output of this process is the variable Intra8x8PredMode[luma8x8BlkIdx].

Table 8-3 specifies the values for Intra8x8PredMode[luma8x8BlkIdx] and the associated mnemonic names.

Table 8-3 – Specification of Intra8x8PredMode[luma8x8BlkIdx] and associated names

Intra8x8PredMode[luma8x8BlkIdx]	Name of Intra8x8PredMode[luma8x8BlkIdx]
0	Intra_8x8_Vertical (prediction mode)
1	Intra_8x8_Horizontal (prediction mode)
2	Intra_8x8_DC (prediction mode)
3	Intra_8x8_Diagonal_Down_Left (prediction mode)
4	Intra_8x8_Diagonal_Down_Right (prediction mode)
5	Intra_8x8_Vertical_Right (prediction mode)
6	Intra_8x8_Horizontal_Down (prediction mode)
7	Intra_8x8_Vertical_Left (prediction mode)
8	Intra_8x8_Horizontal_Up (prediction mode)

Intra8x8PredMode[luma8x8BlkIdx] is derived as follows.

- The process specified in subclause 6.4.8.2 is invoked with luma8x8BlkIdx given as input and the output is assigned to mbAddrA, luma8x8BlkIdxA, mbAddrB, and luma8x8BlkIdxB.
- The variable dcPredModePredictedFlag is derived as follows.
 - If any of the following conditions are true, dcPredModePredictedFlag is set equal to 1
 - the macroblock with address mbAddrA is not available
 - the macroblock with address mbAddrB is not available
 - the macroblock with address mbAddrA is available and coded in Inter prediction mode and constrained_intra_pred_flag is equal to 1

- the macroblock with address mbAddrB is available and coded in Inter prediction mode and constrained_intra_pred_flag is equal to 1
- Otherwise, dcPredModePredictedFlag is set equal to 0.
- For N being either replaced by A or B, the variables intraMxMPredModeN are derived as follows.
 - If dcPredModePredictedFlag is equal to 1 or (the macroblock with address mbAddrN is not coded in Intra_4x4 macroblock prediction mode and the macroblock with address mbAddrN is not coded in Intra_8x8 macroblock prediction mode), intraMxMPredModeN is set equal to 2 (Intra_8x8_DC prediction mode).
 - Otherwise (dcPredModePredictedFlag is equal to 0 and (the macroblock with address mbAddrN is coded in Intra_4x4 macroblock prediction mode or the macroblock with address mbAddrN is coded in Intra_8x8 macroblock prediction mode)), the following applies.
 - If the macroblock with address mbAddrN is coded in Intra_8x8 macroblock mode, intraMxMPredModeN is set equal to Intra8x8PredMode[luma8x8BlkIdxN], where Intra8x8PredMode is the variable array assigned to the macroblock mbAddrN.
 - Otherwise (the macroblock with address mbAddrN is coded in Intra_4x4 macroblock mode), intraMxMPredModeN is derived by the following procedure, where Intra4x4PredMode is the variable array assigned to the macroblock mbAddrN.

$$\text{intraMxMPredModeN} = \text{Intra4x4PredMode}[\text{luma8x8BlkIdxN} * 4 + n] \quad (8-71)$$

where the variable n is derived as follows

- If N is equal to A, depending on the variable MbaffFrameFlag, the variable luma8x8BlkIdx, the current macroblock, and the macroblock mbAddrN, the following applies.
 - If MbaffFrameFlag is equal to 1, the current macroblock is a frame coded macroblock, the macroblock mbAddrN is a field coded macroblock, and luma8x8BlkIdx is equal to 2, n is set equal to 3.
 - Otherwise (MbaffFrameFlag is equal to 0 or the current macroblock is a field coded macroblock or the macroblock mbAddrN is a frame coded macroblock or luma8x8BlkIdx is not equal to 2), n is set equal to 1.
- Otherwise (N is equal to B), n is set equal to 2.
- Finally, given intraMxMPredModeA and intraMxMPredModeB, the variable Intra8x8PredMode[luma8x8BlkIdx] is derived by applying the following procedure.

$$\begin{aligned}
 &\text{predIntra8x8PredMode} = \text{Min}(\text{intraMxMPredModeA}, \text{intraMxMPredModeB}) \\
 &\text{if}(\text{prev_intra8x8_pred_mode_flag}[\text{luma8x8BlkIdx}]) \\
 &\quad \text{Intra8x8PredMode}[\text{luma8x8BlkIdx}] = \text{predIntra8x8PredMode} \\
 &\text{else} \\
 &\quad \text{if}(\text{rem_intra8x8_pred_mode}[\text{luma8x8BlkIdx}] < \text{predIntra8x8PredMode}) \\
 &\quad \quad \text{Intra8x8PredMode}[\text{luma8x8BlkIdx}] = \text{rem_intra8x8_pred_mode}[\text{luma8x8BlkIdx}] \\
 &\quad \text{else} \\
 &\quad \quad \text{Intra8x8PredMode}[\text{luma8x8BlkIdx}] = \text{rem_intra8x8_pred_mode}[\text{luma8x8BlkIdx}] + 1
 \end{aligned} \quad (8-72)$$

8.3.2.2 Intra_8x8 sample prediction

This process is invoked for each 8x8 luma block of a macroblock with prediction mode equal to Intra_8x8 followed by the transform decoding process and picture construction process prior to deblocking for each 8x8 luma block.

Input to this process is the index of an 8x8 luma block luma8x8BlkIdx.

Output of this process are the prediction samples pred8x8L[x, y], with x, y = 0..7 for the 8x8 luma block with index luma8x8BlkIdx.

The position of the upper-left sample of an 8x8 luma block with index luma8x8BlkIdx inside the current macroblock is derived by invoking the inverse 8x8 luma block scanning process in subclause 6.4.4 with luma8x8BlkIdx as the input and the output being assigned to (xO, yO).

The 25 neighbouring samples p[x, y] that are constructed luma samples prior to the deblocking filter process, with x = -1, y = -1..7 and x = 0..15, y = -1, are derived as follows.

- The luma location (xN, yN) is specified by

$$xN = xO + x \quad (8-73)$$

$$yN = yO + y \quad (8-74)$$

- The derivation process for neighbouring locations in subclause 6.4.9 is invoked for luma locations with (xN, yN) as input and mbAddrN and (xW, yW) as output.
- Each sample p[x, y] with x = -1, y = -1..7 and x = 0..15, y = -1 is derived as follows.
 - If any of the following conditions is true, the sample p[x, y] is marked as “not available for Intra_8x8 prediction”
 - mbAddrN is not available,
 - the macroblock mbAddrN is coded in Inter prediction mode and constrained_intra_pred_flag is equal to 1,
 - Otherwise, the sample p[x, y] is marked as “available for Intra_8x8 prediction” and the luma sample at luma location (xW, yW) inside the macroblock mbAddrN is assigned to p[x, y].

When samples p[x, -1], with x = 8..15 are marked as “not available for Intra_8x8 prediction,” and the sample p[7, -1] is marked as “available for Intra_8x8 prediction,” the sample value of p[7, -1] is substituted for sample values p[x, -1], with x = 8..15 and samples p[x, -1], with x = 8..15 are marked as “available for Intra_8x8 prediction”.

NOTE – Each block is assumed to be constructed into a picture array prior to decoding of the next block.

The reference sample filtering process for Intra_8x8 sample prediction in subclause 8.3.2.2.1 is invoked with the samples p[x, y] with x = -1, y = -1..7 and x = 0..15, y = -1 (if available) as input and p'[x, y] with x = -1, y = -1..7 and x = 0..15, y = -1 as output.

Depending on Intra8x8PredMode[luma8x8BlkIdx], one of the Intra_8x8 prediction modes specified in subclauses 8.3.2.2.2 to 8.3.2.2.10 is invoked.

8.3.2.2.1 Reference sample filtering process for Intra_8x8 sample prediction

Inputs to this process are the reference samples p[x, y] with x = -1, y = -1..7 and x = 0..15, y = -1 (if available) for Intra_8x8 sample prediction.

Outputs of this process are the filtered reference samples p'[x, y] with x = -1, y = -1..7 and x = 0..15, y = -1 for Intra_8x8 sample prediction.

When all samples p[x, -1] with x = 0..7 are marked as “available for Intra_8x8 prediction”, the following applies.

- The value of p'[0, -1] is derived as follows.
 - If p[-1, -1] is marked as “available for Intra_8x8 prediction”, p'[0, -1] is derived by

$$p'[0, -1] = (p[-1, -1] + 2 * p[0, -1] + p[1, -1] + 2) >> 2 \quad (8-75)$$

- Otherwise (p[-1, -1] is marked as “not available for Intra_8x8 prediction”), p'[0, -1] is derived by

$$p'[0, -1] = (3 * p[0, -1] + p[1, -1] + 2) >> 2 \quad (8-76)$$

- The values of p'[x, -1], with x = 1..7 are derived by

$$p'[x, -1] = (p[x-1, -1] + 2 * p[x, -1] + p[x+1, -1] + 2) >> 2 \quad (8-77)$$

When all samples p[x, -1] with x = 7..15 are marked as “available for Intra_8x8 prediction”, the following applies.

- The values of p'[x, -1], with x = 8..14 are derived by

$$p'[x, -1] = (p[x-1, -1] + 2 * p[x, -1] + p[x+1, -1] + 2) >> 2 \quad (8-78)$$

- The value of p'[15, -1] is derived by

$$p'[15, -1] = (p[14, -1] + 3 * p[15, -1] + 2) >> 2 \quad (8-79)$$

When the sample $p[-1, -1]$ is marked as “available for Intra_8x8 prediction”, the value of $p'[-1, -1]$ is derived as follows.

- If the sample $p[0, -1]$ is marked as “not available for Intra_8x8 prediction” or the sample $p[-1, 0]$ is marked as “not available for Intra_8x8 prediction”, the following applies.

- If the sample $p[0, -1]$ is marked as “available for Intra_8x8 prediction”, $p'[-1, -1]$ is derived by

$$p'[-1, -1] = (3 * p[-1, -1] + p[0, -1] + 2) >> 2 \quad (8-80)$$

- Otherwise (the sample $p[0, -1]$ is marked as “not available for Intra_8x8 prediction” and the sample $p[-1, 0]$ is marked as “available for Intra_8x8 prediction”), $p'[-1, -1]$ is derived by

$$p'[-1, -1] = (3 * p[-1, -1] + p[-1, 0] + 2) >> 2 \quad (8-81)$$

- Otherwise (the sample $p[0, -1]$ is marked as “available for Intra_8x8 prediction” and the sample $p[-1, 0]$ is marked as “available for Intra_8x8 prediction”), $p'[-1, -1]$ is derived by

$$p'[-1, -1] = (p[0, -1] + 2 * p[-1, -1] + p[-1, 0] + 2) >> 2 \quad (8-82)$$

When all samples $p[-1, y]$ with $y = 0..7$ are marked as “available for Intra_8x8 prediction”, the following applies.

- The value of $p'[-1, 0]$ is derived as follows.

- If $p[-1, -1]$ is marked as “available for Intra_8x8 prediction”, $p'[-1, 0]$ is derived by

$$p'[-1, 0] = (p[-1, -1] + 2 * p[-1, 0] + p[-1, 1] + 2) >> 2 \quad (8-83)$$

- Otherwise (if $p[-1, -1]$ is marked as “not available for Intra_8x8 prediction”), $p'[-1, 0]$ is derived by

$$p'[-1, 0] = (3 * p[-1, 0] + p[-1, 1] + 2) >> 2 \quad (8-84)$$

- The values of $p'[-1, y]$, with $y = 1..6$ are derived by

$$p'[-1, y] = (p[-1, y-1] + 2 * p[-1, y] + p[-1, y+1] + 2) >> 2 \quad (8-85)$$

- The value of $p'[-1, 7]$ is derived by

$$p'[-1, 7] = (p[-1, 6] + 3 * p[-1, 7] + 2) >> 2 \quad (8-86)$$

8.3.2.2.2 Specification of Intra_8x8_Veritical prediction mode

This Intra_8x8 prediction mode is invoked when $\text{Intra8x8PredMode}[\text{luma8x8BlkIdx}]$ is equal to 0.

This mode shall be used only when the samples $p[x, -1]$ with $x = 0..7$ are marked as “available for Intra_8x8 prediction”.

The values of the prediction samples $\text{pred8x8}_L[x, y]$, with $x, y = 0..7$ are derived by

$$\text{pred8x8}_L[x, y] = p'[x, -1], \text{ with } x, y = 0..7 \quad (8-87)$$

8.3.2.2.3 Specification of Intra_8x8_Horizontal prediction mode

This Intra_8x8 prediction mode is invoked when $\text{Intra8x8PredMode}[\text{luma8x8BlkIdx}]$ is equal to 1.

This mode shall be used only when the samples $p[-1, y]$, with $y = 0..7$ are marked as “available for Intra_8x8 prediction”.

The values of the prediction samples $\text{pred8x8}_L[x, y]$, with $x, y = 0..7$ are derived by

$$\text{pred8x8}_L[x, y] = p'[-1, y], \text{ with } x, y = 0..7 \quad (8-88)$$

8.3.2.2.4 Specification of Intra_8x8_DC prediction mode

This Intra_8x8 prediction mode is invoked when Intra8x8PredMode[luma8x8BlkIdx] is equal to 2.

The values of the prediction samples $\text{pred8x8}_L[x, y]$, with $x, y = 0..7$ are derived as follows.

- If all samples $p[x, -1]$, with $x = 0..7$ and $p[-1, y]$, with $y = 0..7$ are marked as “available for Intra_8x8 prediction,” the values of the prediction samples $\text{pred8x8}_L[x, y]$, with $x, y = 0..7$ are derived by

$$\text{pred8x8}_L[x, y] = \left(\sum_{x'=0}^7 p'[x', -1] + \sum_{y'=0}^7 p'[-1, y'] + 8 \right) \gg 4 \quad (8-89)$$

- Otherwise, if any samples $p[x, -1]$, with $x = 0..7$ are marked as “not available for Intra_8x8 prediction” and all samples $p[-1, y]$, with $y = 0..7$ are marked as “available for Intra_8x8 prediction”, the values of the prediction samples $\text{pred8x8}_L[x, y]$, with $x, y = 0..7$ are derived by

$$\text{pred8x8}_L[x, y] = \left(\sum_{y'=0}^7 p'[-1, y'] + 4 \right) \gg 3 \quad (8-90)$$

- Otherwise, if any samples $p[-1, y]$, with $y = 0..7$ are marked as “not available for Intra_8x8 prediction” and all samples $p[x, -1]$, with $x = 0..7$ are marked as “available for Intra_8x8 prediction”, the values of the prediction samples $\text{pred8x8}_L[x, y]$, with $x, y = 0..7$ are derived by

$$\text{pred8x8}_L[x, y] = \left(\sum_{x'=0}^7 p'[x', -1] + 4 \right) \gg 3 \quad (8-91)$$

- Otherwise (some samples $p[x, -1]$, with $x = 0..7$ and some samples $p[-1, y]$, with $y = 0..7$ are marked as “not available for Intra_8x8 prediction”), the values of the prediction samples $\text{pred8x8}_L[x, y]$, with $x, y = 0..7$ are derived by

$$\text{pred8x8}_L[x, y] = (1 \ll (\text{BitDepth}_Y - 1)) \quad (8-92)$$

NOTE – An 8x8 luma block can always be predicted using this mode.

8.3.2.2.5 Specification of Intra_8x8_Diagonal_Down_Left prediction mode

This Intra_8x8 prediction mode is invoked when Intra8x8PredMode[luma8x8BlkIdx] is equal to 3.

This mode shall be used only when the samples $p[x, -1]$ with $x = 0..15$ are marked as “available for Intra_8x8 prediction”.

The values of the prediction samples $\text{pred8x8}_L[x, y]$, with $x, y = 0..7$ are derived as follows.

- If x is equal to 7 and y is equal to 7,

$$\text{pred8x8}_L[x, y] = (p[14, -1] + 3 * p[15, -1] + 2) \gg 2 \quad (8-93)$$

- Otherwise (x is not equal to 7 or y is not equal to 7),

$$\text{pred8x8}_L[x, y] = (p[x + y, -1] + 2 * p[x + y + 1, -1] + p[x + y + 2, -1] + 2) \gg 2 \quad (8-94)$$

8.3.2.2.6 Specification of Intra_8x8_Diagonal_Down_Right prediction mode

This Intra_8x8 prediction mode is invoked when Intra8x8PredMode[luma8x8BlkIdx] is equal to 4.

This mode shall be used only when the samples $p[x, -1]$ with $x = 0..7$ and $p[-1, y]$ with $y = -1..7$ are marked as “available for Intra_8x8 prediction”.

The values of the prediction samples $\text{pred8x8}_L[x, y]$, with $x, y = 0..7$ are derived as follows.

- If x is greater than y ,

$$\text{pred8x8}_L[x, y] = (p[x - y - 2, -1] + 2 * p[x - y - 1, -1] + p[x - y, -1] + 2) \gg 2 \quad (8-95)$$

- Otherwise if x is less than y,

$$\text{pred8x8}_L[x, y] = (p'[-1, y - x - 2] + 2 * p'[-1, y - x - 1] + p'[-1, y - x] + 2) \gg 2 \quad (8-96)$$

- Otherwise (x is equal to y),

$$\text{pred8x8}_L[x, y] = (p'[0, -1] + 2 * p'[-1, -1] + p'[-1, 0] + 2) \gg 2 \quad (8-97)$$

8.3.2.2.7 Specification of Intra_8x8_Vertical_Right prediction mode

This Intra_8x8 prediction mode is invoked when Intra8x8PredMode[luma8x8BlkIdx] is equal to 5.

This mode shall be used only when the samples $p[x, -1]$ with $x = 0..7$ and $p[-1, y]$ with $y = -1..7$ are marked as “available for Intra_8x8 prediction”.

Let the variable zVR be set equal to $2 * x - y$.

The values of the prediction samples $\text{pred8x8}_L[x, y]$, with $x, y = 0..7$ are derived as follows.

- If zVR is equal to 0, 2, 4, 6, 8, 10, 12, or 14

$$\text{pred8x8}_L[x, y] = (p'[x - (y \gg 1) - 1, -1] + p'[x - (y \gg 1), -1] + 1) \gg 1 \quad (8-98)$$

- Otherwise, if zVR is equal to 1, 3, 5, 7, 9, 11, or 13

$$\text{pred8x8}_L[x, y] = (p'[x - (y \gg 1) - 2, -1] + 2 * p'[x - (y \gg 1) - 1, -1] + p'[x - (y \gg 1), -1] + 2) \gg 2 \quad (8-99)$$

- Otherwise, if zVR is equal to -1,

$$\text{pred8x8}_L[x, y] = (p'[-1, 0] + 2 * p'[-1, -1] + p'[0, -1] + 2) \gg 2 \quad (8-100)$$

- Otherwise (zVR is equal to -2, -3, -4, -5, -6, or -7),

$$\text{pred8x8}_L[x, y] = (p'[-1, y - 2*x - 1] + 2 * p'[-1, y - 2*x - 2] + p'[-1, y - 2*x - 3] + 2) \gg 2 \quad (8-101)$$

8.3.2.2.8 Specification of Intra_8x8_Horizontal_Down prediction mode

This Intra_8x8 prediction mode is invoked when Intra8x8PredMode[luma8x8BlkIdx] is equal to 6.

This mode shall be used only when the samples $p[x, -1]$ with $x = 0..7$ and $p[-1, y]$ with $y = -1..7$ are marked as “available for Intra_8x8 prediction”.

Let the variable zHD be set equal to $2 * y - x$.

The values of the prediction samples $\text{pred8x8}_L[x, y]$, with $x, y = 0..7$ are derived as follows.

- If zHD is equal to 0, 2, 4, 6, 8, 10, 12, or 14

$$\text{pred8x8}_L[x, y] = (p'[-1, y - (x \gg 1) - 1] + p'[-1, y - (x \gg 1)] + 1) \gg 1 \quad (8-102)$$

- Otherwise, if zHD is equal to 1, 3, 5, 7, 9, 11, or 13

$$\text{pred8x8}_L[x, y] = (p'[-1, y - (x \gg 1) - 2] + 2 * p'[-1, y - (x \gg 1) - 1] + p'[-1, y - (x \gg 1)] + 2) \gg 2 \quad (8-103)$$

- Otherwise, if zHD is equal to -1,

$$\text{pred8x8}_L[x, y] = (p'[-1, 0] + 2 * p'[-1, -1] + p'[0, -1] + 2) \gg 2 \quad (8-104)$$

- Otherwise (zHD is equal to -2, -3, -4, -5, -6, -7),

$$\text{pred8x8}_L[x, y] = (p'[x - 2*y - 1, -1] + 2 * p'[x - 2*y - 2, -1] + p'[x - 2*y - 3, -1] + 2) \gg 2 \quad (8-105)$$

8.3.2.2.9 Specification of Intra_8x8_Vertical_Left prediction mode

This Intra_8x8 prediction mode is invoked when Intra8x8PredMode[luma8x8BlkIdx] is equal to 7.

This mode shall be used only when the samples $p[x, -1]$ with $x = 0..15$ are marked as “available for Intra_8x8 prediction”.

The values of the prediction samples $\text{pred8x8}_L[x, y]$, with $x, y = 0..7$ are derived as follows.

- If y is equal to 0, 2, 4 or 6

$$\text{pred8x8}_L[x, y] = (p'[x + (y >> 1), -1] + p'[x + (y >> 1) + 1, -1] + 1) >> 1 \quad (8-106)$$

- Otherwise (y is equal to 1, 3, 5, 7),

$$\text{pred8x8}_L[x, y] = (p'[x + (y >> 1), -1] + 2 * p'[x + (y >> 1) + 1, -1] + p'[x + (y >> 1) + 2, -1] + 2) >> 2 \quad (8-107)$$

8.3.2.2.10 Specification of Intra_8x8_Horizontal_Up prediction mode

This Intra_8x8 prediction mode is invoked when Intra8x8PredMode[luma8x8BlkIdx] is equal to 8.

This mode shall be used only when the samples $p[-1, y]$ with $y = 0..7$ are marked as “available for Intra_8x8 prediction”.

Let the variable zHU be set equal to $x + 2 * y$.

The values of the prediction samples $\text{pred8x8}_L[x, y]$, with $x, y = 0..7$ are derived as follows:

- If zHU is equal to 0, 2, 4, 6, 8, 10, or 12

$$\text{pred8x8}_L[x, y] = (p'[-1, y + (x >> 1)] + p'[-1, y + (x >> 1) + 1] + 1) >> 1 \quad (8-108)$$

- Otherwise, if zHU is equal to 1, 3, 5, 7, 9, or 11

$$\text{pred8x8}_L[x, y] = (p'[-1, y + (x >> 1)] + 2 * p'[-1, y + (x >> 1) + 1] + p'[-1, y + (x >> 1) + 2] + 2) >> 2 \quad (8-109)$$

- Otherwise, if zHU is equal to 13,

$$\text{pred8x8}_L[x, y] = (p'[-1, 6] + 3 * p'[-1, 7] + 2) >> 2 \quad (8-110)$$

- Otherwise (zHU is greater than 13),

$$\text{pred8x8}_L[x, y] = p'[-1, 7] \quad (8-111)$$

8.3.3 Intra_16x16 prediction process for luma samples

This process is invoked when the macroblock prediction mode is equal to Intra_16x16. It specifies how the Intra prediction luma samples for the current macroblock are derived.

Outputs of this process are Intra prediction luma samples for the current macroblock $\text{pred}_L[x, y]$.

The 33 neighbouring samples $p[x, y]$ that are constructed luma samples prior to the deblocking filter process, with $x = -1, y = -1..15$ and with $x = 0..15, y = -1$, are derived as follows.

- The derivation process for neighbouring locations in subclause 6.4.9 is invoked for luma locations with (x, y) assigned to (xN, yN) as input and mbAddrN and (xW, yW) as output.
- Each sample $p[x, y]$ with $x = -1, y = -1..15$ and with $x = 0..15, y = -1$ is derived as follows.
 - If any of the following conditions is true, the sample $p[x, y]$ is marked as “not available for Intra_16x16 prediction”
 - mbAddrN is not available,
 - the macroblock mbAddrN is coded in Inter prediction mode and $\text{constrained_intra_pred_flag}$ is equal to 1.

- the macroblock mbAddrN has mb_type equal to SI and constrained_intra_pred_flag is equal to 1.
- Otherwise, the sample $p[x, y]$ is marked as “available for Intra_16x16 prediction” and the luma sample at luma location (xW, yW) inside the macroblock mbAddrN is assigned to $p[x, y]$.

Let $\text{pred}_L[x, y]$ with $x, y = 0..15$ denote the prediction samples for the 16x16 luma block samples.

Intra_16x16 prediction modes are specified in Table 8-4.

Table 8-4 – Specification of Intra16x16PredMode and associated names

Intra16x16PredMode	Name of Intra16x16PredMode
0	Intra_16x16_Vertical (prediction mode)
1	Intra_16x16_Horizontal (prediction mode)
2	Intra_16x16_DC (prediction mode)
3	Intra_16x16_Plane (prediction mode)

Depending on Intra16x16PredMode, one of the Intra_16x16 prediction modes specified in subclauses 8.3.3.1 to 8.3.3.4 is invoked.

8.3.3.1 Specification of Intra_16x16_Vertical prediction mode

This Intra_16x16 prediction mode shall be used only when the samples $p[x, -1]$ with $x = 0..15$ are marked as “available for Intra_16x16 prediction”.

$$\text{pred}_L[x, y] = p[x, -1], \text{ with } x, y = 0..15 \quad (8-112)$$

8.3.3.2 Specification of Intra_16x16_Horizontal prediction mode

This Intra_16x16 prediction mode shall be used only when the samples $p[-1, y]$ with $y = 0..15$ are marked as “available for Intra_16x16 prediction”.

$$\text{pred}_L[x, y] = p[-1, y], \text{ with } x, y = 0..15 \quad (8-113)$$

8.3.3.3 Specification of Intra_16x16_DC prediction mode

This Intra_16x16 prediction mode operates, depending on whether the neighbouring samples are marked as “available for Intra_16x16 prediction”, as follows.

- If all neighbouring samples $p[x, -1]$, with $x = 0..15$ and $p[-1, y]$, with $y = 0..15$ are marked as “available for Intra_16x16 prediction”, the prediction for all luma samples in the macroblock is given by:

$$\text{pred}_L[x, y] = \left(\sum_{x'=0}^{15} p[x', -1] + \sum_{y'=0}^{15} p[-1, y'] + 16 \right) \gg 5, \text{ with } x, y = 0..15 \quad (8-114)$$

- Otherwise, if any of the neighbouring samples $p[x, -1]$, with $x = 0..15$ are marked as "not available for Intra_16x16 prediction" and all of the neighbouring samples $p[-1, y]$, with $y = 0..15$ are marked as “available for Intra_16x16 prediction”, the prediction for all luma samples in the macroblock is given by:

$$\text{pred}_L[x, y] = \left(\sum_{y'=0}^{15} p[-1, y'] + 8 \right) \gg 4, \text{ with } x, y = 0..15 \quad (8-115)$$

- Otherwise, if any of the neighbouring samples $p[-1, y]$, with $y = 0..15$ are marked as "not available for Intra_16x16 prediction" and all of the neighbouring samples $p[x, -1]$, with $x = 0..15$ are marked as “available for Intra_16x16 prediction”, the prediction for all luma samples in the macroblock is given by:

$$\text{pred}_L[x, y] = \left(\sum_{x'=0}^{15} p[x', -1] + 8 \right) \gg 4, \text{ with } x, y = 0..15 \quad (8-116)$$

- Otherwise (some of the neighbouring samples $p[x, -1]$, with $x = 0..15$ and some of the neighbouring samples $p[-1, y]$, with $y = 0..15$ are marked as “not available for Intra_16x16 prediction”), the prediction for all luma samples in the macroblock is given by:

$$\text{pred}_L[x, y] = (1 \ll (\text{BitDepth}_Y - 1)), \text{ with } x, y = 0..15 \quad (8-117)$$

8.3.3.4 Specification of Intra_16x16_Plane prediction mode

This Intra_16x16 prediction mode shall be used only when the samples $p[x, -1]$ with $x = -1..15$ and $p[-1, y]$ with $y = 0..15$ are marked as “available for Intra_16x16 prediction”.

$$\text{pred}_L[x, y] = \text{Clip1}_Y((a + b * (x - 7) + c * (y - 7) + 16) \gg 5), \text{ with } x, y = 0..15, \quad (8-118)$$

where:

$$a = 16 * (p[-1, 15] + p[15, -1]) \quad (8-119)$$

$$b = (5 * H + 32) \gg 6 \quad (8-120)$$

$$c = (5 * V + 32) \gg 6 \quad (8-121)$$

and H and V are specified in Equations 8-122 and 8-123.

$$H = \sum_{x'=0}^7 (x'+1) * (p[8+x', -1] - p[6-x', -1]) \quad (8-122)$$

$$V = \sum_{y'=0}^7 (y'+1) * (p[-1, 8+y'] - p[-1, 6-y']) \quad (8-123)$$

8.3.4 Intra prediction process for chroma samples

This process is invoked for I and SI macroblock types. It specifies how the Intra prediction chroma samples for the current macroblock are derived.

Outputs of this process are Intra prediction chroma samples for the current macroblock $\text{pred}_{Cb}[x, y]$ and $\text{pred}_{Cr}[x, y]$.

Both chroma blocks (Cb and Cr) of the macroblock use the same prediction mode. The prediction mode is applied to each of the chroma blocks separately. The process specified in this subclause is invoked for each chroma block. In the remainder of this subclause, chroma block refers to one of the two chroma blocks and the subscript C is used as a replacement of the subscript Cb or Cr.

The neighbouring samples $p[x, y]$ that are constructed chroma samples prior to the deblocking filter process, with $x = -1, y = -1..MbHeightC - 1$ and with $x = 0..MbWidthC - 1, y = -1$, are derived as follows.

- The derivation process for neighbouring locations in subclause 6.4.9 is invoked for chroma locations with (x, y) assigned to (xN, yN) as input and $mbAddrN$ and (xW, yW) as output.
- Each sample $p[x, y]$ is derived as follows.
 - If any of the following conditions is true, the sample $p[x, y]$ is marked as “not available for Intra chroma prediction”
 - $mbAddrN$ is not available,
 - the macroblock $mbAddrN$ is coded in Inter prediction mode and $\text{constrained_intra_pred_flag}$ is equal to 1.
 - the macroblock $mbAddrN$ has mb_type equal to SI and $\text{constrained_intra_pred_flag}$ is equal to 1 and the current macroblock does not have mb_type equal to SI.
 - Otherwise, the sample $p[x, y]$ is marked as “available for Intra chroma prediction” and the chroma sample of component C at chroma location (xW, yW) inside the macroblock $mbAddrN$ is assigned to $p[x, y]$.

Let $\text{pred}_C[x, y]$ with $x = 0..MbWidthC - 1$, $y = 0..MbHeightC - 1$ denote the prediction samples for the chroma block samples.

Intra chroma prediction modes are specified in Table 8-5.

Table 8-5 – Specification of Intra chroma prediction modes and associated names

intra_chroma_pred_mode	Name of intra_chroma_pred_mode
0	Intra_Chroma_DC (prediction mode)
1	Intra_Chroma_Horizontal (prediction mode)
2	Intra_Chroma_Vertical (prediction mode)
3	Intra_Chroma_Plane (prediction mode)

Depending on `intra_chroma_pred_mode`, one of the Intra chroma prediction modes specified in subclauses 8.3.4.1 to 8.3.4.4 is invoked.

8.3.4.1 Specification of Intra_Chroma_DC prediction mode

This Intra chroma prediction mode is invoked when `intra_chroma_pred_mode` is equal to 0.

For each chroma block of 4x4 samples indexed by `chroma4x4BlkIdx = 0..(1 << (chroma_format_idc + 1)) - 1`, the following applies.

- Depending on `chroma_format_idc`, the position of the upper-left sample of a 4x4 chroma block with index `chroma4x4BlkIdx` is derived as follows

- If `chroma_format_idc` is equal to 1 or 2, the following applies

$$xO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 0) \quad (8-124)$$

$$yO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 1) \quad (8-125)$$

- Otherwise (`chroma_format_idc` is equal to 3), the following applies

$$xO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx} / 4, 8, 8, 16, 0) + \text{InverseRasterScan}(\text{chroma4x4BlkIdx} \% 4, 4, 4, 8, 0) \quad (8-126)$$

$$yO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx} / 4, 8, 8, 16, 1) + \text{InverseRasterScan}(\text{chroma4x4BlkIdx} \% 4, 4, 4, 8, 1) \quad (8-127)$$

- If (xO, yO) is equal to $(0, 0)$ or xO and yO are greater than 0, the values of the prediction samples $\text{pred}_C[x + xO, y + yO]$ with $x, y = 0..3$ are derived as follows.

- If all samples $p[x + xO, -1]$, with $x = 0..3$ and $p[-1, y + yO]$, with $y = 0..3$ are marked as “available for Intra chroma prediction”, the values of the prediction samples $\text{pred}_C[x + xO, y + yO]$, with $x, y = 0..3$ are derived as

$$\text{pred}_C[x + xO, y + yO] = \left(\sum_{x'=0}^3 p[x' + xO, -1] + \sum_{y'=0}^3 p[-1, y' + yO] + 4 \right) >> 3, \text{ with } x, y = 0..3. \quad (8-128)$$

- Otherwise, if any samples $p[x + xO, -1]$, with $x = 0..3$ are marked as “not available for Intra chroma prediction” and all samples $p[-1, y + yO]$, with $y = 0..3$ are marked as “available for Intra chroma prediction”, the values of the prediction samples $\text{pred}_C[x + xO, y + yO]$, with $x, y = 0..3$ are derived as

$$\text{pred}_C[x + xO, y + yO] = \left(\sum_{y'=0}^3 p[-1, y' + yO] + 2 \right) >> 2, \text{ with } x, y = 0..3. \quad (8-129)$$

- Otherwise, if any samples $p[-1, y+yO]$, with $y = 0..3$ are marked as “not available for Intra chroma prediction” and all samples $p[x+xO, -1]$, with $x = 0..3$ are marked as “available for Intra chroma prediction”, the values of the prediction samples $pred_C[x+xO, y+yO]$, with $x, y = 0..3$ are derived as

$$pred_C[x+xO, y+yO] = \left(\sum_{x'=0}^3 p[x'+xO, -1] + 2 \right) \gg 2, \text{ with } x, y = 0..3. \quad (8-130)$$

- Otherwise (some samples $p[x+xO, -1]$, with $x = 0..3$ and some samples $p[-1, y+yO]$, with $y = 0..3$ are marked as “not available for Intra chroma prediction”), the values of the prediction samples $pred_C[x+xO, y+yO]$, with $x, y = 0..3$ are derived as

$$pred_C[x+xO, y+yO] = (1 \ll (\text{BitDepth}_C - 1)), \text{ with } x, y = 0..3. \quad (8-131)$$

- Otherwise, if xO is greater than 0 and yO is equal to 0, the values of the prediction samples $pred_C[x+xO, y+yO]$ with $x, y = 0..3$ are derived as follows.

- If all samples $p[x+xO, -1]$, with $x = 0..3$ are marked as “available for Intra chroma prediction”, the values of the prediction samples $pred_C[x+xO, y+yO]$, with $x, y = 0..3$ are derived as

$$pred_C[x+xO, y+yO] = \left(\sum_{x'=0}^3 p[x'+xO, -1] + 2 \right) \gg 2, \text{ with } x, y = 0..3. \quad (8-132)$$

- Otherwise, if all samples $p[-1, y+yO]$, with $y = 0..3$ are marked as “available for Intra chroma prediction”, the values of the prediction samples $pred_C[x+xO, y+yO]$, with $x, y = 0..3$ are derived as

$$pred_C[x+xO, y+yO] = \left(\sum_{y'=0}^3 p[-1, y'+yO] + 2 \right) \gg 2, \text{ with } x, y = 0..3. \quad (8-133)$$

- Otherwise (some samples $p[x+xO, -1]$, with $x = 0..3$ and some samples $p[-1, y+yO]$, with $y = 0..3$ are marked as “not available for Intra chroma prediction”), the values of the prediction samples $pred_C[x+xO, y+yO]$, with $x, y = 0..3$ are derived as

$$pred_C[x+xO, y+yO] = (1 \ll (\text{BitDepth}_C - 1)), \text{ with } x, y = 0..3. \quad (8-134)$$

- Otherwise (xO is equal to 0 and yO is greater than 0), the values of the prediction samples $pred_C[x+xO, y+yO]$ with $x, y = 0..3$ are derived as follows.

- If all samples $p[-1, y+yO]$, with $y = 0..3$ are marked as “available for Intra chroma prediction”, the values of the prediction samples $pred_C[x+xO, y+yO]$, with $x, y = 0..3$ are derived as

$$pred_C[x+xO, y+yO] = \left(\sum_{y'=0}^3 p[-1, y'+yO] + 2 \right) \gg 2, \text{ with } x, y = 0..3. \quad (8-135)$$

- Otherwise, if all samples $p[x+xO, -1]$, with $x = 0..3$ are marked as “available for Intra chroma prediction”, the values of the prediction samples $pred_C[x+xO, y+yO]$, with $x, y = 0..3$ are derived as

$$pred_C[x+xO, y+yO] = \left(\sum_{x'=0}^3 p[x'+xO, -1] + 2 \right) \gg 2, \text{ with } x, y = 0..3. \quad (8-136)$$

- Otherwise (some samples $p[x+xO, -1]$, with $x = 0..3$ and some samples $p[-1, y+yO]$, with $y = 0..3$ are marked as “not available for Intra chroma prediction”), the values of the prediction samples $pred_C[x+xO, y+yO]$, with $x, y = 0..3$ are derived as

$$pred_C[x+xO, y+yO] = (1 \ll (\text{BitDepth}_C - 1)), \text{ with } x, y = 0..3. \quad (8-137)$$

8.3.4.2 Specification of Intra_Chroma_Horizontal prediction mode

This Intra chroma prediction mode is invoked when `intra_chroma_pred_mode` is equal to 1.

This mode shall be used only when the samples `p[-1, y]` with `y = 0..MbHeightC - 1` are marked as "available for Intra chroma prediction".

The values of the prediction samples `predC[x, y]` are derived as follows.

$$\text{pred}_C[x, y] = p[-1, y], \text{ with } x = 0..MbWidthC - 1 \text{ and } y = 0..MbHeightC - 1 \quad (8-138)$$

8.3.4.3 Specification of Intra_Chroma_Vertical prediction mode

This Intra chroma prediction mode is invoked when `intra_chroma_pred_mode` is equal to 2.

This mode shall be used only when the samples `p[x, -1]` with `x = 0..MbWidthC - 1` are marked as "available for Intra chroma prediction".

The values of the prediction samples `predC[x, y]` are derived as follows.

$$\text{pred}_C[x, y] = p[x, -1], \text{ with } x = 0..MbWidthC - 1 \text{ and } y = 0..MbHeightC - 1 \quad (8-139)$$

8.3.4.4 Specification of Intra_Chroma_Plane prediction mode

This Intra chroma prediction mode is invoked when `intra_chroma_pred_mode` is equal to 3.

This mode shall be used only when the samples `p[x, -1]`, with `x = 0..MbWidthC - 1` and `p[-1, y]`, with `y = -1..MbHeightC - 1` are marked as "available for Intra chroma prediction".

The values of the prediction samples `predC[x, y]` are derived as follows.

Let the variable `xCF` be set equal to `4 * (chroma_format_idc == 3)` and let the variable `yCF` be set equal to `4 * (chroma_format_idc != 1)`.

$$\text{pred}_C[x, y] = \text{Clip1}_C((a + b * (x - 3 - xCF) + c * (y - 3 - yCF) + 16) \gg 5), \\ \text{with } x = 0..MbWidthC - 1 \text{ and } y = 0..MbHeightC - 1 \quad (8-140)$$

where:

$$a = 16 * (p[-1, MbHeightC - 1] + p[MbWidthC - 1, -1]) \quad (8-141)$$

$$b = ((34 - 29 * (chroma_format_idc == 3)) * H + 32) \gg 6 \quad (8-142)$$

$$c = ((34 - 29 * (chroma_format_idc != 1)) * V + 32) \gg 6 \quad (8-143)$$

and `H` and `V` are specified as

$$H = \sum_{x'=0}^{3+xCF} (x'+1) * (p[4+xCF+x', -1] - p[2+xCF-x', -1]) \quad (8-144)$$

$$V = \sum_{y'=0}^{3+yCF} (y'+1) * (p[-1, 4+yCF+y'] - p[-1, 2+yCF-y']) \quad (8-145)$$

8.3.5 Sample construction process for I_PCM macroblocks

This process is invoked when `mb_type` is equal to `I_PCM`.

The variable `dy` is derived as follows.

- If `MbaffFrameFlag` is equal to 1 and the current macroblock is a field macroblock, `dy` is set equal to 2.
- Otherwise (`MbaffFrameFlag` is equal to 0 or the current macroblock is a frame macroblock), `dy` is set equal to 1.

The position of the upper-left luma sample of the current macroblock is derived by invoking the inverse macroblock scanning process in subclause 6.4.1 with CurrMbAddr as input and the output being assigned to (xP, yP).

The constructed luma samples prior to the deblocking process are generated as specified by:

$$\text{for}(i = 0; i < 256; i++) \\ S'_{L}[xP + (i \% 16), yP + dy * (i / 16)] = \text{pcm_sample_luma}[i] \quad (8-146)$$

When chroma_format_idc is not equal to 0 (monochrome), the constructed chroma samples prior to the deblocking process are generated as specified by:

$$\begin{aligned} &\text{for}(i = 0; i < \text{MbWidthC} * \text{MbHeightC}; i++) \{ \\ &\quad S'_{Cb}[(xP / \text{SubWidthC}) + (i \% \text{MbWidthC}), \\ &\quad \quad ((yP + \text{SubHeightC} - 1) / \text{SubHeightC}) + dy * (i / \text{MbWidthC})] = \\ &\quad \quad \text{pcm_sample_chroma}[i] \\ &\quad S'_{Cr}[(xP / \text{SubWidthC}) + (i \% \text{MbWidthC}), \\ &\quad \quad ((yP + \text{SubHeightC} - 1) / \text{SubHeightC}) + dy * (i / \text{MbWidthC})] = \\ &\quad \quad \text{pcm_sample_chroma}[i + \text{MbWidthC} * \text{MbHeightC}] \\ &\} \end{aligned} \quad (8-147)$$

8.4 Inter prediction process

This process is invoked when decoding P and B macroblock types.

Outputs of this process are Inter prediction samples for the current macroblock that are a 16x16 array pred_L of luma samples and when chroma_format_idc is not equal to 0 (monochrome) two 8x8 arrays pred_{Cr} and pred_{Cb} of chroma samples, one for each of the chroma components Cb and Cr.

The partitioning of a macroblock is specified by mb_type. Each macroblock partition is referred to by mbPartIdx. When the macroblock partitioning consists of partitions that are equal to sub-macroblocks, each sub-macroblock can be further partitioned into sub-macroblock partitions as specified by sub_mb_type. Each sub-macroblock partition is referred to by subMbPartIdx. When the macroblock partitioning does not consist of sub-macroblocks, subMbPartIdx is set equal to 0.

The following steps are specified for each macroblock partition or for each sub-macroblock partition.

The functions MbPartWidth(), MbPartHeight(), SubMbPartWidth(), and SubMbPartHeight() describing the width and height of macroblock partitions and sub-macroblock partitions are specified in Tables 7-13, 7-14, 7-17, and 7-18.

The range of the macroblock partition index mbPartIdx is derived as follows.

- If mb_type is equal to B_Skip or B_Direct_16x16, mbPartIdx proceeds over values 0..3.
- Otherwise (mb_type is not equal to B_Skip or B_Direct_16x16), mbPartIdx proceeds over values 0..NumMbPart(mb_type) – 1.

For each value of mbPartIdx, the variables partWidth and partHeight for each macroblock partition or sub-macroblock partition in the macroblock are derived as follows.

- If mb_type is not equal to P_8x8, P_8x8ref0, B_Skip, B_Direct_16x16, or B_8x8, subMbPartIdx is set equal to 0, and partWidth and partHeight are derived as

$$\text{partWidth} = \text{MbPartWidth}(\text{mb_type}) \quad (8-148)$$

$$\text{partHeight} = \text{MbPartHeight}(\text{mb_type}) \quad (8-149)$$

- Otherwise, if mb_type is equal to P_8x8 or P_8x8ref0, or mb_type is equal to B_8x8 and sub_mb_type[mbPartIdx] is not equal to B_Direct_8x8, subMbPartIdx proceeds over values 0..NumSubMbPart(sub_mb_type) – 1, and partWidth and partHeight are derived as

$$\text{partWidth} = \text{SubMbPartWidth}(\text{sub_mb_type}[\text{mbPartIdx}]) \quad (8-150)$$

$$\text{partHeight} = \text{SubMbPartHeight}(\text{sub_mb_type}[\text{mbPartIdx}]). \quad (8-151)$$

- Otherwise (mb_type is equal to B_Skip or B_Direct_16x16, or mb_type is equal to B_8x8 and sub_mb_type[mbPartIdx] is equal to B_Direct_8x8), subMbPartIdx proceeds over values 0..3, and partWidth and partHeight are derived as

$$\text{partWidth} = 4 \quad (8-152)$$

$$\text{partHeight} = 4 \quad (8-153)$$

When chroma_format_idc is not equal to 0 (monochrome) the variables partWidthC and partHeightC are derived as

$$\text{partWidthC} = \text{partWidth} / \text{SubWidthC} \quad (8-154)$$

$$\text{partHeightC} = \text{partHeight} / \text{SubHeightC} \quad (8-155)$$

Let the variable MvCnt be initially set equal to 0 before any invocation of subclause 8.4.1 for the macroblock.

The Inter prediction process for a macroblock partition mbPartIdx and a sub-macroblock partition subMbPartIdx consists of the following ordered steps

1. Derivation process for motion vector components and reference indices as specified in subclause 8.4.1.

Inputs to this process are

- a macroblock partition mbPartIdx,
- a sub-macroblock partition subMbPartIdx.

Outputs of this process are

- luma motion vectors mvL0 and mvL1 and when chroma_format_idc is not equal to 0 (monochrome) the chroma motion vectors mvCL0 and mvCL1
- reference indices refIdxL0 and refIdxL1
- prediction list utilization flags predFlagL0 and predFlagL1
- the sub-macroblock partition motion vector count subMvCnt.

2. The variable MvCnt is incremented by subMvCnt.

3. Decoding process for Inter prediction samples as specified in subclause 8.4.2.

Inputs to this process are

- a macroblock partition mbPartIdx,
- a sub-macroblock partition subMbPartIdx.
- variables specifying partition width and height for luma and chroma (if available), partWidth, partHeight, partWidthC (if available), and partHeightC (if available)
- luma motion vectors mvL0 and mvL1 and when chroma_format_idc is not equal to 0 (monochrome) the chroma motion vectors mvCL0 and mvCL1
- reference indices refIdxL0 and refIdxL1
- prediction list utilization flags predFlagL0 and predFlagL1

Outputs of this process are

- inter prediction samples (pred); which are a (partWidth)x(partHeight) array predPart_L of prediction luma samples and when chroma_format_idc is not equal to 0 (monochrome) two (partWidthC)x(partHeightC) arrays predPart_{Cb}, and predPart_{Cr} of prediction chroma samples, one for each of the chroma components Cb and Cr.

For use in derivation processes of variables invoked later in the decoding process, the following assignments are made:

$$\text{MvL0}[\text{mbPartIdx}][\text{subMbPartIdx}] = \text{mvL0} \quad (8-156)$$

$$\text{MvL1}[\text{mbPartIdx}][\text{subMbPartIdx}] = \text{mvL1} \quad (8-157)$$

$$\text{RefIdxL0}[\text{mbPartIdx}] = \text{refIdxL0} \quad (8-158)$$

$$\text{RefIdxL1}[\text{mbPartIdx}] = \text{refIdxL1} \quad (8-159)$$

$$\text{PredFlagL0}[\text{mbPartIdx}] = \text{predFlagL0} \quad (8-160)$$

$$\text{PredFlagL1}[\text{mbPartIdx}] = \text{predFlagL1} \quad (8-161)$$

The location of the upper-left sample of the partition relative to the upper-left sample of the macroblock is derived by invoking the inverse macroblock partition scanning process as described in subclause 6.4.2.1 with `mbPartIdx` as the input and `(xP, yP)` as the output.

The location of the upper-left sample of the macroblock sub-partition relative to the upper-left sample of the macroblock partition is derived by invoking the inverse sub-macroblock partition scanning process as described in subclause 6.4.2.2 with `subMbPartIdx` as the input and `(xS, yS)` as the output.

The macroblock prediction is formed by placing the partition or sub-macroblock partition prediction samples in their correct relative positions in the macroblock, as follows.

The variable `predL[xP + xS + x, yP + yS + y]` with `x = 0 .. partWidth - 1`, `y = 0 .. partHeight - 1` is derived by

$$\text{predL}[\text{xP} + \text{xS} + \text{x}, \text{yP} + \text{yS} + \text{y}] = \text{predPartL}[\text{x}, \text{y}] \quad (8-162)$$

When `chroma_format_idc` is not equal to 0 (monochrome) the variable `predC` with `x = 0..partWidthC - 1`, `y = 0..partHeightC - 1`, and C in `predC` and `predPartC` being replaced by Cb or Cr is derived by

$$\text{predC}[\text{xP} / \text{SubWidthC} + \text{xS} / \text{SubWidthC} + \text{x}, \text{yP} / \text{SubHeightC} + \text{yS} / \text{SubHeightC} + \text{y}] = \text{predPartC}[\text{x}, \text{y}] \quad (8-163)$$

8.4.1 Derivation process for motion vector components and reference indices

Inputs to this process are

- a macroblock partition `mbPartIdx`,
- a sub-macroblock partition `subMbPartIdx`.

Outputs of this process are

- luma motion vectors `mvL0` and `mvL1` as well as the chroma motion vectors `mvCL0` and `mvCL1`
- reference indices `refIdxL0` and `refIdxL1`
- prediction list utilization flags `predFlagL0` and `predFlagL1`
- a sub-partition macroblock motion vector count variable `subMvCnt`

For the derivation of the variables `mvL0` and `mvL1` as well as `refIdxL0` and `refIdxL1`, the following applies.

- If `mb_type` is equal to `P_Skip`, the derivation process for luma motion vectors for skipped macroblocks in P and SP slices in subclause 8.4.1.1 is invoked with the output being the luma motion vectors `mvL0` and reference indices `refIdxL0`, and `predFlagL0` is set equal to 1. `mvL1` and `refIdxL1` are marked as not available and `predFlagL1` is set equal to 0. The sub-partition motion vector count variable `subMvCnt` is set equal to 1.
- Otherwise, if `mb_type` is equal to `B_Skip` or `B_Direct_16x16` or `sub_mb_type[mbPartIdx]` is equal to `B_Direct_8x8`, the derivation process for luma motion vectors for `B_Skip`, `B_Direct_16x16`, and `B_Direct_8x8` in B slices in subclause 8.4.1.2 is invoked with `mbPartIdx` and `subMbPartIdx` as the input and the output being the luma motion vectors `mvL0`, `mvL1`, the reference indices `refIdxL0`, `refIdxL1`, the sub-partition motion vector count `subMvCnt`, and the prediction utilization flags `predFlagL0` and `predFlagL1`.
- Otherwise, for X being replaced by either 0 or 1 in the variables `predFlagLX`, `mvLX`, `refIdxLX`, and in `Pred_LX` and in the syntax elements `ref_idx_IX` and `mvd_IX`, the following applies.

1. The variables refIdxLX and predFlagLX are derived as follows.

- If MbPartPredMode(mb_type, mbPartIdx) or SubMbPredMode(sub_mb_type[mbPartIdx]) is equal to Pred_LX or to BiPred,

$$\text{refIdxLX} = \text{ref_idx_lX}[\text{mbPartIdx}] \quad (8-164)$$

$$\text{predFlagLX} = 1 \quad (8-165)$$

- Otherwise, the variables refIdxLX and predFlagLX are specified by

$$\text{refIdxLX} = -1 \quad (8-166)$$

$$\text{predFlagLX} = 0 \quad (8-167)$$

2. The variable subMvCnt for sub-partition motion vector count is set equal to predFlagL0 + predFlagL1.

3. The variable currSubMbType is derived as follows.

- If the macroblock type is equal to B_8x8, currSubMbType is set equal to sub_mb_type[mbPartIdx].
- Otherwise (the macroblock type is not equal to B_8x8), currSubMbType is set equal to "na".

4. When predFlagLX is equal to 1, the derivation process for luma motion vector prediction in subclause 8.4.1.3 is invoked with mbPartIdx, subMbPartIdx, refIdxLX, and currSubMbType as the inputs and the output being mvLX. The luma motion vectors are derived by

$$\text{mvLX}[0] = \text{mvpLX}[0] + \text{mvd_lX}[\text{mbPartIdx}][\text{subMbPartIdx}][0] \quad (8-168)$$

$$\text{mvLX}[1] = \text{mvpLX}[1] + \text{mvd_lX}[\text{mbPartIdx}][\text{subMbPartIdx}][1] \quad (8-169)$$

For the derivation of the variables for the chroma motion vectors, the following applies. When predFlagLX (with X being either 0 or 1) is equal to 1, the derivation process for chroma motion vectors in subclause 8.4.1.4 is invoked with mvLX and refIdxLX as input and the output being mvCLX.

8.4.1.1 Derivation process for luma motion vectors for skipped macroblocks in P and SP slices

This process is invoked when mb_type is equal to P_Skip.

Outputs of this process are the motion vector mvL0 and the reference index refIdxL0.

The reference index refIdxL0 for a skipped macroblock is derived as follows.

$$\text{refIdxL0} = 0. \quad (8-170)$$

For the derivation of the motion vector mvL0 of a P_Skip macroblock type, the following applies.

- The process specified in subclause 8.4.1.3.2 is invoked with mbPartIdx set equal to 0, subMbPartIdx set equal to 0, currSubMbType set equal to "na", and listSuffixFlag set equal to 0 as input and the output is assigned to mbAddrA, mbAddrB, mvL0A, mvL0B, refIdxL0A, and refIdxL0B.
- The variable mvL0 is specified as follows.
 - If any of the following conditions are true, both components of the motion vector mvL0 are set equal to 0.
 - mbAddrA is not available
 - mbAddrB is not available
 - refIdxL0A is equal to 0 and both components of mvL0A are equal to 0
 - refIdxL0B is equal to 0 and both components of mvL0B are equal to 0

- Otherwise, the derivation process for luma motion vector prediction as specified in subclause 8.4.1.3 is invoked with `mbPartIdx = 0`, `subMbPartIdx = 0`, `refIdxL0`, and `currSubMbType = "na"` as inputs and the output is assigned to `mvL0`.

NOTE – The output is directly assigned to `mvL0`, since the predictor is equal to the actual motion vector.

8.4.1.2 Derivation process for luma motion vectors for B_Skip, B_Direct_16x16, and B_Direct_8x8

This process is invoked when `mb_type` is equal to `B_Skip` or `B_Direct_16x16`, or `sub_mb_type[mbPartIdx]` is equal to `B_Direct_8x8`.

Inputs to this process are `mbPartIdx` and `subMbPartIdx`.

Outputs of this process are the reference indices `refIdxL0`, `refIdxL1`, the motion vectors `mvL0` and `mvL1`, the sub-partition motion vector count `subMvCnt`, and the prediction list utilization flags, `predFlagL0` and `predFlagL1`.

The derivation process depends on the value of `direct_spatial_mv_pred_flag`, which is present in the bitstream in the slice header syntax as specified in subclause 7.3.3, and is specified as follows.

- If `direct_spatial_mv_pred_flag` is equal to 1, the mode in which the outputs of this process are derived is referred to as spatial direct prediction mode.
- Otherwise (`direct_spatial_mv_pred_flag` is equal to 0), mode in which the outputs of this process are derived is referred to as temporal direct prediction mode.

Both spatial and temporal direct prediction mode use the co-located motion vectors and reference indices as specified in subclause 8.4.1.2.1.

The motion vectors and reference indices are derived as follows.

- If spatial direct prediction mode is used, the direct motion vector and reference index prediction mode specified in subclause 8.4.1.2.2 is used, with `subMvCnt` being an output.
- Otherwise (temporal direct prediction mode is used), the direct motion vector and reference index prediction mode specified in subclause 8.4.1.2.3 is used and the variable `subMvCnt` is derived as follows.
 - If `subMbPartIdx` is equal to 0, `subMvCnt` is set equal to 2.
 - Otherwise (`subMbPartIdx` is not equal to 0), `subMvCnt` is set equal to 0.

8.4.1.2.1 Derivation process for the co-located 4x4 sub-macroblock partitions

Inputs to this process are `mbPartIdx` and `subMbPartIdx`.

Outputs of this process are the picture `colPic`, the co-located macroblock `mbAddrCol`, the motion vector `mvCol`, the reference index `refIdxCol`, and the variable `vertMvScale` (which can be `One_To_One`, `Frm_To_Fld` or `Fld_To_Frm`).

When `RefPicList1[0]` is a frame or a complementary field pair, let `firstRefPicL1Top` and `firstRefPicL1Bottom` be the top and bottom fields of `RefPicList1[0]`, respectively, and let the following variables be specified as

$$\text{topAbsDiffPOC} = \text{Abs}(\text{DiffPicOrderCnt}(\text{firstRefPicL1Top}, \text{CurrPic})) \quad (8-171)$$

$$\text{bottomAbsDiffPOC} = \text{Abs}(\text{DiffPicOrderCnt}(\text{firstRefPicL1Bottom}, \text{CurrPic})) \quad (8-172)$$

The variable colPic specifies the picture that contains the co-located macroblock as specified in Table 8-6.

Table 8-6 – Specification of the variable colPic

field_pic_flag	RefPicList1[0] is ...	mb_field_decoding_flag	additional condition	colPic
1	a field of a decoded frame			the frame containing RefPicList1[0]
	a decoded field			RefPicList1[0]
0	a decoded frame			RefPicList1[0]
	a complementary field pair	0	topAbsDiffPOC < bottomAbsDiffPOC	firstRefPicL1Top
			topAbsDiffPOC >= bottomAbsDiffPOC	firstRefPicL1Bottom
		1	(CurrMbAddr & 1) == 0	firstRefPicL1Top
			(CurrMbAddr & 1) != 0	firstRefPicL1Bottom

When direct_8x8_inference_flag is equal to 1, subMbPartIdx is set as follows.

$$\text{subMbPartIdx} = \text{mbPartIdx} \quad (8-173)$$

Let PicCodingStruct(X) be a function with the argument X being either CurrPic or colPic. It is specified in Table 8-7.

Table 8-7 – Specification of PicCodingStruct(X)

X is coded with field_pic_flag equal to ...	mb_adaptive_frame_field_flag	PicCodingStruct(X)
1		FLD
0	0	FRM
0	1	AFRM

With luma4x4BlkIdx = mbPartIdx * 4 + subMbPartIdx, the inverse 4x4 luma block scanning process as specified in subclause 6.4.3 is invoked with luma4x4BlkIdx as the input and (x, y) assigned to (xCol, yCol) as the output.

Table 8-8 specifies the co-located macroblock address mbAddrCol, yM, and the variable vertMvScale in two steps:

1. Specification of a macroblock address mbAddrX depending on PicCodingStruct(CurrPic), and PicCodingStruct(colPic).

NOTE – It is not possible for CurrPic and colPic picture coding types to be either (FRM, AFRM) or (AFRM, FRM) because these picture coding types must be separated by an IDR picture.

2. Specification of mbAddrCol, yM, and vertMvScale depending on mb_field_decoding_flag and the variable fieldDecodingFlagX, which is derived as follows.
 - If the macroblock mbAddrX in the picture colPic is a field macroblock, fieldDecodingFlagX is set equal to 1
 - Otherwise (the macroblock mbAddrX in the picture colPic is a frame macroblock), fieldDecodingFlagX is set equal to 0.

Unspecified values in Table 8-8 indicate that the value of the corresponding variable is not relevant for the current table row.

mbAddrCol is set equal to CurrMbAddr or to one of the following values.

$$\text{mbAddrCol1} = 2 * \text{PicWidthInMbs} * (\text{CurrMbAddr} / \text{PicWidthInMbs}) + (\text{CurrMbAddr} \% \text{PicWidthInMbs}) + \text{PicWidthInMbs} * (\text{yCol} / 8) \quad (8-174)$$

$$\text{mbAddrCol2} = 2 * \text{CurrMbAddr} + (\text{yCol} / 8) \quad (8-175)$$

$$\text{mbAddrCol3} = 2 * \text{CurrMbAddr} + \text{bottom_field_flag} \quad (8-176)$$

$$\text{mbAddrCol4} = \text{PicWidthInMbs} * (\text{CurrMbAddr} / (2 * \text{PicWidthInMbs})) + (\text{CurrMbAddr} \% \text{PicWidthInMbs}) \quad (8-177)$$

$$\text{mbAddrCol5} = \text{CurrMbAddr} / 2 \quad (8-178)$$

$$\text{mbAddrCol6} = 2 * (\text{CurrMbAddr} / 2) + ((\text{topAbsDiffPOC} < \text{bottomAbsDiffPOC}) ? 0 : 1) \quad (8-179)$$

$$\text{mbAddrCol7} = 2 * (\text{CurrMbAddr} / 2) + (\text{yCol} / 8) \quad (8-180)$$

Table 8-8 – Specification of mbAddrCol, yM, and vertMvScale

PicCodingStruct(CurrPic)	PicCodingStruct(colPic)	mbAddrX	mb_field_decoding_flag	fieldDecodingFlagX	mbAddrCol	yM	vertMvScale
FLD	FLD				CurrMbAddr	yCol	One_To_One
	FRM				mbAddrCol1	(2 * yCol) % 16	Frm_To_Fld
	AFRM	2*CurrMbAddr	0		mbAddrCol2	(2 * yCol) % 16	Frm_To_Fld
			1		mbAddrCol3	yCol	One_To_One
FRM	FLD				mbAddrCol4	8 * ((CurrMbAddr / PicWidthInMbs) % 2) + 4 * (yCol / 8)	Fld_To_Frm
	FRM				CurrMbAddr	yCol	One_To_One
AFRM	FLD		0		mbAddrCol5	8 * (CurrMbAddr % 2) + 4 * (yCol / 8)	Fld_To_Frm
			1		mbAddrCol5	yCol	One_To_One
	AFRM	CurrMbAddr	0	0	CurrMbAddr	yCol	One_To_One
			1		mbAddrCol6	8 * (CurrMbAddr % 2) + 4 * (yCol / 8)	Fld_To_Frm
		CurrMbAddr	0		mbAddrCol7	(2 * yCol) % 16	Frm_To_Fld
			1		CurrMbAddr	yCol	One_To_One

Let mbPartIdxCol be the macroblock partition index of the co-located partition and subMbPartIdxCol the sub-macroblock partition index of the co-located sub-macroblock partition. The partition in the macroblock mbAddrCol inside the picture colPic covering the sample (xCol, yM) is assigned to mbPartIdxCol and the sub-macroblock partition inside the partition mbPartIdxCol covering the sample (xCol, yM) in the macroblock mbAddrCol inside the picture colPic is assigned to subMbPartIdxCol.

The prediction utilization flags `predFlagL0Col` and `predFlagL1Col` are set equal to `PredFlagL0[mbPartIdxCol]` and `PredFlagL1[mbPartIdxCol]`, respectively, which are the prediction utilization flags that have been assigned to the macroblock partition `mbAddrCol\mbPartIdxCol` inside the picture `colPic`.

The motion vector `mvCol` and the reference index `refIdxCol` are derived as follows.

- If the macroblock `mbAddrCol` is coded in Intra macroblock prediction mode or both prediction utilization flags, `predFlagL0Col` and `predFlagL1Col` are equal to 0, both components of `mvCol` are set equal to 0 and `refIdxCol` is set equal to -1.
- Otherwise, the following applies.
 - If `predFlagL0Col` is equal to 1, the motion vector `mvCol` and the reference index `refIdxCol` are set equal to `MvL0[mbPartIdxCol][subMbPartIdxCol]` and `RefIdxL0[mbPartIdxCol]`, respectively, which are the motion vector `mvL0` and the reference index `refIdxL0` that have been assigned to the (sub-)macroblock partition `mbAddrCol\mbPartIdxCol\subMbPartIdxCol` inside the picture `colPic`.
 - Otherwise (`predFlagL0Col` is equal to 0 and `predFlagL1Col` is equal to 1), the motion vector `mvCol` and the reference index `refIdxCol` are set equal to `MvL1[mbPartIdxCol][subMbPartIdxCol]` and `RefIdxL1[mbPartIdxCol]`, respectively, which are the motion vector `mvL1` and the reference index `refIdxL1` that have been assigned to the (sub-)macroblock partition `mbAddrCol\mbPartIdxCol\subMbPartIdxCol` inside the picture `colPic`.

8.4.1.2.2 Derivation process for spatial direct luma motion vector and reference index prediction mode

This process is invoked when `direct_spatial_mv_pred_flag` is equal to 1 and any of the following conditions is true.

- `mb_type` is equal to `B_Skip`
- `mb_type` is equal to `B_Direct_16x16`
- `sub_mb_type[mbPartIdx]` is equal to `B_Direct_8x8`.

Inputs to this process are `mbPartIdx`, `subMbPartIdx`.

Outputs of this process are the reference indices `refIdxL0`, `refIdxL1`, the motion vectors `mvL0` and `mvL1`, the sub-partition motion vector count `subMvCnt`, and the prediction list utilization flags, `predFlagL0` and `predFlagL1`.

The reference indices `refIdxL0` and `refIdxL1` and the variable `directZeroPredictionFlag` are derived by applying the following ordered steps.

1. Let the variable `currSubMbType` be set equal to `sub_mb_type[mbPartIdx]`.
2. The process specified in subclause 8.4.1.3.2 is invoked with `mbPartIdx = 0`, `subMbPartIdx = 0`, `currSubMbType`, and `listSuffixFlag = 0` as inputs and the output is assigned to the motion vectors `mvL0N` and the reference indices `refIdxL0N` with `N` being replaced by `A`, `B`, or `C`.
3. The process specified in subclause 8.4.1.3.2 is invoked with `mbPartIdx = 0`, `subMbPartIdx = 0`, `currSubMbType`, and `listSuffixFlag = 1` as inputs and the output is assigned to the motion vectors `mvL1N` and the reference indices `refIdxL1N` with `N` being replaced by `A`, `B`, or `C`.

NOTE 1 – The motion vectors `mvL0N`, `mvL1N` and the reference indices `refIdxL0N`, `refIdxL1N` are identical for all 4x4 sub-macroblock partitions of a macroblock.

4. The reference indices `refIdxL0`, `refIdxL1`, and `directZeroPredictionFlag` are derived by

$$\text{refIdxL0} = \text{MinPositive}(\text{refIdxL0A}, \text{MinPositive}(\text{refIdxL0B}, \text{refIdxL0C})) \quad (8-181)$$

$$\text{refIdxL1} = \text{MinPositive}(\text{refIdxL1A}, \text{MinPositive}(\text{refIdxL1B}, \text{refIdxL1C})) \quad (8-182)$$

$$\text{directZeroPredictionFlag} = 0 \quad (8-183)$$

where

$$\text{MinPositive}(x, y) = \begin{cases} \text{Min}(x, y) & \text{if } x \geq 0 \text{ and } y \geq 0 \\ \text{Max}(x, y) & \text{otherwise} \end{cases} \quad (8-184)$$

5. When both reference indices refIdxL0 and refIdxL1 are less than 0,

$$\text{refIdxL0} = 0 \quad (8-185)$$

$$\text{refIdxL1} = 0 \quad (8-186)$$

$$\text{directZeroPredictionFlag} = 1 \quad (8-187)$$

The process specified in subclause 8.4.1.2.1 is invoked with mbPartIdx, subMbPartIdx given as input and the output is assigned to refIdxCol and mvCol.

The variable colZeroFlag is derived as follows.

- If all of the following conditions are true, colZeroFlag is set equal to 1.
 - RefPicList1[0] is currently marked as "used for short-term reference".
 - refIdxCol is equal to 0
 - both motion vector components mvCol[0] and mvCol[1] lie in the range of -1 to 1 in units specified as follows.
 - If the co-located macroblock is a frame macroblock, the units of mvCol[0] and mvCol[1] are units of quarter luma frame samples.
 - Otherwise (the co-located macroblock is a field macroblock), the units of mvCol[0] and mvCol[1] are units of quarter luma field samples.

NOTE 2 – For purposes of determining the condition above, the value mvCol[1] is not scaled to use the units of a motion vector for the current macroblock in cases when the current macroblock is a frame macroblock and the co-located macroblock is a field macroblock or when the current macroblock is a field macroblock and the co-located macroblock is a frame macroblock. This aspect differs from the use of mvCol[1] in the temporal direct mode as specified in subclause 8.4.1.2.3, which applies scaling to the motion vector of the co-located macroblock to use the same units as the units of a motion vector for the current macroblock, using Equation 8-190 or Equation 8-191 in these cases.

- Otherwise, colZeroFlag is set equal to 0.

The motion vectors mvLX (with X being 0 or 1) are derived as follows.

- If any of the following conditions is true, both components of the motion vector mvLX are set equal to 0.
 - directZeroPredictionFlag is equal to 1
 - refIdxLX is less than 0
 - refIdxLX is equal to 0 and colZeroFlag is equal to 1
- Otherwise, the process specified in subclause 8.4.1.3 is invoked with mbPartIdx = 0, subMbPartIdx = 0, refIdxLX, and currSubMbType as inputs and the output is assigned to mvLX.

NOTE 3 – The motion vector mvLX returned from subclause 8.4.1.3 is identical for all 4x4 sub-macroblock partitions of a macroblock for which the process is invoked.

The prediction utilization flags predFlagL0 and predFlagL1 are derived as specified using Table 8-9.

Table 8-9 – Assignment of prediction utilization flags

refIdxL0	refIdxL1	predFlagL0	predFlagL1
≥ 0	≥ 0	1	1
≥ 0	< 0	1	0
< 0	≥ 0	0	1

The variable subMvCnt is derived as follows.

- If subMbPartIdx is not equal to 0 or direct_8x8_inference_flag is equal to 0, subMvCnt is set equal to 0.
- Otherwise (subMbPartIdx is equal to 0 and direct_8x8_inference_flag is equal to 1), subMvCnt is set equal to predFlagL0 + predFlagL1.

8.4.1.2.3 Derivation process for temporal direct luma motion vector and reference index prediction mode

This process is invoked when `direct_spatial_mv_pred_flag` is equal to 0 and any of the following conditions is true.

- `mb_type` is equal to `B_Skip`
- `mb_type` is equal to `B_Direct_16x16`
- `sub_mb_type[mbPartIdx]` is equal to `B_Direct_8x8`.

Inputs to this process are `mbPartIdx` and `subMbPartIdx`.

Outputs of this process are the motion vectors `mvL0` and `mvL1`, the reference indices `refIdxL0` and `refIdxL1`, and the prediction list utilization flags, `predFlagL0` and `predFlagL1`.

The process specified in subclause 8.4.1.2.1 is invoked with `mbPartIdx`, `subMbPartIdx` given as input and the output is assigned to `colPic`, `mbAddrCol`, `mvCol`, `refIdxCol`, and `vertMvScale`.

The reference indices `refIdxL0` and `refIdxL1` are derived as follows.

$$\text{refIdxL0} = ((\text{refIdxCol} < 0) ? 0 : \text{MapColToList0}(\text{refIdxCol})) \quad (8-188)$$

$$\text{refIdxL1} = 0 \quad (8-189)$$

NOTE 1 – If the current macroblock is a field macroblock, `refIdxL0` and `refIdxL1` index a list of fields; otherwise (the current macroblock is a frame macroblock), `refIdxL0` and `refIdxL1` index a list of frames or complementary reference field pairs.

Let `refPicCol` be a frame, a field, or a complementary field pair that was referred by the reference index `refIdxCol` when decoding the co-located macroblock `mbAddrCol` inside the picture `colPic`. The function `MapColToList0(refIdxCol)` is specified as follows.

- If `vertMvScale` is equal to `One_To_One`, the following applies.
 - If `field_pic_flag` is equal to 0 and the current macroblock is a field macroblock, the following applies.
 - Let `refIdxL0Frm` be the lowest valued reference index in the current reference picture list `RefPicList0` that references the frame or complementary field pair that contains the field `refPicCol`. `RefPicList0` shall contain a frame or complementary field pair that contains the field `refPicCol`. The return value of `MapColToList0()` is specified as follows.
 - If the field referred to by `refIdxCol` has the same parity as the current macroblock, `MapColToList0(refIdxCol)` returns the reference index $(\text{refIdxL0Frm} \ll 1)$.
 - Otherwise (the field referred by `refIdxCol` has the opposite parity of the current macroblock), `MapColToList0(refIdxCol)` returns the reference index $((\text{refIdxL0Frm} \ll 1) + 1)$.
 - Otherwise (`field_pic_flag` is equal to 1 or the current macroblock is a frame macroblock), `MapColToList0(refIdxCol)` returns the lowest valued reference index `refIdxL0` in the current reference picture list `RefPicList0` that references `refPicCol`. `RefPicList0` shall contain `refPicCol`.
 - Otherwise, if `vertMvScale` is equal to `Frm_To_Fld`, the following applies.
 - If `field_pic_flag` is equal to 0, let `refIdxL0Frm` be the lowest valued reference index in the current reference picture list `RefPicList0` that references `refPicCol`. `MapColToList0(refIdxCol)` returns the reference index $(\text{refIdxL0Frm} \ll 1)$. `RefPicList0` shall contain `refPicCol`.
 - Otherwise (`field_pic_flag` is equal to 1), `MapColToList0(refIdxCol)` returns the lowest valued reference index `refIdxL0` in the current reference picture list `RefPicList0` that references the field of `refPicCol` with the same parity as the current picture `CurrPic`. `RefPicList0` shall contain the field of `refPicCol` with the same parity as the current picture `CurrPic`.
 - Otherwise (`vertMvScale` is equal to `Fld_To_Frm`), `MapColToList0(refIdxCol)` returns the lowest valued reference index `refIdxL0` in the current reference picture list `RefPicList0` that references the frame or complementary field pair that contains `refPicCol`. `RefPicList0` shall contain a frame or complementary field pair that contains the field `refPicCol`.

NOTE 2 – A decoded reference picture that was marked as "used for short-term reference" when it was referenced in the decoding process of the picture containing the co-located macroblock may have been modified to be marked as "used for long-term reference" before being used for reference for inter prediction using the direct prediction mode for the current macroblock.

Depending on the value of `vertMvScale` the vertical component of `mvCol` is modified as follows.

- If `vertMvScale` is equal to `Frm_To_Fld`

$$\text{mvCol}[1] = \text{mvCol}[1] / 2 \quad (8-190)$$

- Otherwise, if `vertMvScale` is equal to `Fld_To_Frm`

$$\text{mvCol}[1] = \text{mvCol}[1] * 2 \quad (8-191)$$

- Otherwise (`vertMvScale` is equal to `One_To_One`), `mvCol[1]` remains unchanged.

The variables `currPicOrField`, `pic0`, and `pic1`, are derived as follows.

- If `field_pic_flag` is equal to 0 and the current macroblock is a field macroblock, the following applies.
 - `currPicOrField` is the field of the current picture `CurrPic` that has the same parity as the current macroblock.
 - `pic1` is the field of `RefPicList1[0]` that has the same parity as the current macroblock.
 - The variable `pic0` is derived as follows.
 - If `refIdxL0 % 2` is equal to 0, `pic0` is the field of `RefPicList0[refIdxL0 / 2]` that has the same parity as the current macroblock.
 - Otherwise (`refIdxL0 % 2` is not equal to 0), `pic0` is the field of `RefPicList0[refIdxL0 / 2]` that has the opposite parity of the current macroblock.
- Otherwise (`field_pic_flag` is equal to 1 or the current macroblock is a frame macroblock), `currPicOrField` is the current picture `CurrPic`, `pic1` is the decoded reference picture `RefPicList1[0]`, and `pic0` is the decoded reference picture `RefPicList0[refIdxL0]`.

The two motion vectors `mvL0` and `mvL1` for each 4x4 sub-macroblock partition of the current macroblock are derived as follows:

NOTE 3 – It is often the case that many of the 4x4 sub-macroblock partitions share the same motion vectors and reference pictures. In these cases, temporal direct mode motion compensation can calculate the inter prediction sample values in larger units than 4x4 luma sample blocks. For example, when `direct_8x8_inference_flag` is equal to 1, at least each 8x8 luma sample quadrant of the macroblock shares the same motion vectors and reference pictures.

- If the reference index `refIdxL0` refers to a long-term reference picture, or `DiffPicOrderCnt(pic1, pic0)` is equal to 0, the motion vectors `mvL0`, `mvL1` for the direct mode partition are derived by

$$\text{mvL0} = \text{mvCol} \quad (8-192)$$

$$\text{mvL1} = 0 \quad (8-193)$$

- Otherwise, the motion vectors `mvL0`, `mvL1` are derived as scaled versions of the motion vector `mvCol` of the co-located sub-macroblock partition as specified below (see Figure 8-2)

$$\text{tx} = (16384 + \text{Abs}(\text{td} / 2)) / \text{td} \quad (8-194)$$

$$\text{DistScaleFactor} = \text{Clip3}(-1024, 1023, (\text{tb} * \text{tx} + 32) \gg 6) \quad (8-195)$$

$$\text{mvL0} = (\text{DistScaleFactor} * \text{mvCol} + 128) \gg 8 \quad (8-196)$$

$$\text{mvL1} = \text{mvL0} - \text{mvCol} \quad (8-197)$$

where `tb` and `td` are derived as follows.

$$\text{tb} = \text{Clip3}(-128, 127, \text{DiffPicOrderCnt}(\text{currPicOrField}, \text{pic0})) \quad (8-198)$$

$$\text{td} = \text{Clip3}(-128, 127, \text{DiffPicOrderCnt}(\text{pic1}, \text{pic0})) \quad (8-199)$$

NOTE 4 – mvL0 and mvL1 cannot exceed the ranges specified in Annex A.

The prediction utilization flags predFlagL0 and predFlagL1 are both set equal to 1.

Figure 8-2 illustrates the temporal direct-mode motion vector inference when the current picture is temporally between the reference picture from reference picture list 0 and the reference picture from reference picture list 1.

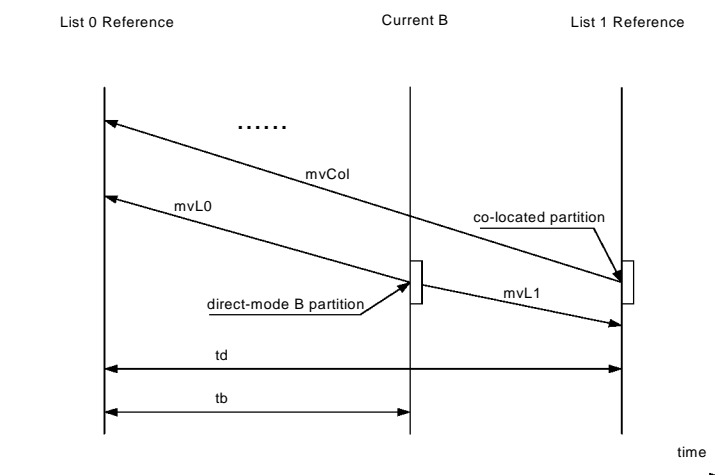


Figure 8-2 – Example for temporal direct-mode motion vector inference (informative)

8.4.1.3 Derivation process for luma motion vector prediction

Inputs to this process are

- the macroblock partition index mbPartIdx,
- the sub-macroblock partition index subMbPartIdx,
- the reference index of the current partition refIdxLX (with X being 0 or 1),
- the variable currSubMbType.

Output of this process is the prediction mvpLX of the motion vector mvLX (with X being 0 or 1).

The derivation process for the neighbouring blocks for motion data in subclause 8.4.1.3.2 is invoked with mbPartIdx, subMbPartIdx, currSubMbType, and listSuffixFlag = X (with X being 0 or 1 for refIdxLX being refIdxL0 or refIdxL1, respectively) as the input and with mbAddrN\mbPartIdxN\subMbPartIdxN, reference indices refIdxLXN and the motion vectors mvLXN with N being replaced by A, B, or C as the output.

The derivation process for median luma motion vector prediction in subclause 8.4.1.3.1 is invoked with mbAddrN\mbPartIdxN\subMbPartIdxN, mvLXN, refIdxLXN with N being replaced by A, B, or C and refIdxLX as the input and mvpLX as the output, unless one of the following is true.

- MbPartWidth(mb_type) is equal to 16, MbPartHeight(mb_type) is equal to 8, mbPartIdx is equal to 0, and refIdxLXB is equal to refIdxLX,

$$\text{mvpLX} = \text{mvLXB} \quad (8-200)$$

- MbPartWidth(mb_type) is equal to 16, MbPartHeight(mb_type) is equal to 8, mbPartIdx is equal to 1, and refIdxLXA is equal to refIdxLX,

$$\text{mvpLX} = \text{mvLXA} \quad (8-201)$$

- MbPartWidth(mb_type) is equal to 8, MbPartHeight(mb_type) is equal to 16, mbPartIdx is equal to 0, and refIdxLXA is equal to refIdxLX,

$$\text{mvpLX} = \text{mvLXA} \quad (8-202)$$

- MbPartWidth(mb_type) is equal to 8, MbPartHeight(mb_type) is equal to 16, mbPartIdx is equal to 1, and refIdxLXC is equal to refIdxLX,

$$\text{mvpLX} = \text{mvLXC} \quad (8-203)$$

Figure 8-3 illustrates the non-median prediction as described above.

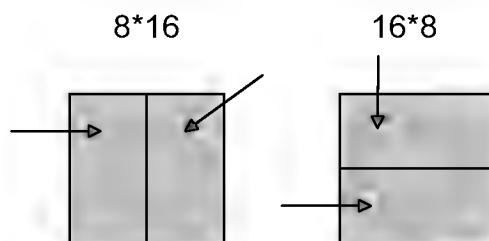


Figure 8-3 – Directional segmentation prediction (informative)

8.4.1.3.1 Derivation process for median luma motion vector prediction

Inputs to this process are

- the neighbouring partitions mbAddrN\mbPartIdxN\subMbPartIdxN (with N being replaced by A, B, or C),
- the motion vectors mvLXN (with N being replaced by A, B, or C) of the neighbouring partitions,
- the reference indices refIdxLXN (with N being replaced by A, B, or C) of the neighbouring partitions, and
- the reference index refIdxLX of the current partition.

Output of this process is the motion vector prediction mvpLX.

The variable mvpLX is derived as follows:

- When both partitions mbAddrB\mbPartIdxB\subMbPartIdxB and mbAddrC\mbPartIdxC\subMbPartIdxC are not available and mbAddrA\mbPartIdxA\subMbPartIdxA is available,

$$\text{mvLXB} = \text{mvLXA} \quad (8-204)$$

$$\text{mvLXC} = \text{mvLXA} \quad (8-205)$$

$$\text{refIdxLXB} = \text{refIdxLXA} \quad (8-206)$$

$$\text{refIdxLXC} = \text{refIdxLXA} \quad (8-207)$$

- Depending on reference indices refIdxLXA, refIdxLXB, or refIdxLXC, the following applies.
 - If one and only one of the reference indices refIdxLXA, refIdxLXB, or refIdxLXC is equal to the reference index refIdxLX of the current partition, the following applies. Let refIdxLXN be the reference index that is equal to refIdxLX, the motion vector mvLXN is assigned to the motion vector prediction mvpLX:

$$\text{mvpLX} = \text{mvLXN} \quad (8-208)$$

- Otherwise, each component of the motion vector prediction mvpLX is given by the median of the corresponding vector components of the motion vector mvLXA, mvLXB, and mvLXC:

$$\text{mvpLX}[0] = \text{Median}(\text{mvLXA}[0], \text{mvLXB}[0], \text{mvLXC}[0]) \quad (8-209)$$

$$\text{mvLX}[1] = \text{Median}(\text{mvLXA}[1], \text{mvLXB}[1], \text{mvLXC}[1]) \quad (8-210)$$

8.4.1.3.2 Derivation process for motion data of neighbouring partitions

Inputs to this process are

- the macroblock partition index mbPartIdx,
- the sub-macroblock partition index subMbPartIdx,
- the current sub-macroblock type currSubMbType,
- the list suffix flag listSuffixFlag

Outputs of this process are (with N being replaced by A, B, or C)

- mbAddrN\mbPartIdxN\subMbPartIdxN specifying neighbouring partitions,
- the motion vectors mvLXN of the neighbouring partitions, and
- the reference indices refIdxLXN of the neighbouring partitions.

Variable names that include the string "LX" are interpreted with the X being equal to listSuffixFlag.

The partitions mbAddrN\mbPartIdxN\subMbPartIdxN with N being either A, B, or C are derived in the following ordered steps.

1. Let mbAddrD\mbPartIdxD\subMbPartIdxD be variables specifying an additional neighbouring partition.
2. The process in subclause 6.4.8.5 is invoked with mbPartIdx, currSubMbType, and subMbPartIdx as input and the output is assigned to mbAddrN\mbPartIdxN\subMbPartIdxN with N being replaced by A, B, C, or D.
3. When the partition mbAddrC\mbPartIdxC\subMbPartIdxC is not available, the following applies

$$\text{mbAddrC} = \text{mbAddrD} \quad (8-211)$$

$$\text{mbPartIdxC} = \text{mbPartIdxD} \quad (8-212)$$

$$\text{subMbPartIdxC} = \text{subMbPartIdxD} \quad (8-213)$$

The motion vectors mvLXN and reference indices refIdxLXN (with N being A, B, or C) are derived as follows.

- If the macroblock partition or sub-macroblock partition mbAddrN\mbPartIdxN\subMbPartIdxN is not available or mbAddrN is coded in Intra prediction mode or predFlagLX of mbAddrN\mbPartIdxN\subMbPartIdxN is equal to 0, both components of mvLXN are set equal to 0 and refIdxLXN is set equal to -1.
- Otherwise, the following applies.
 - The motion vector mvLXN and reference index refIdxLXN are set equal to $\text{MvLX}[\text{mbPartIdxN}][\text{subMbPartIdxN}]$ and $\text{RefIdxLX}[\text{mbPartIdxN}]$, respectively, which are the motion vector mvLX and reference index refIdxLX that have been assigned to the (sub-)macroblock partition mbAddrN\mbPartIdxN\subMbPartIdxN.
 - The variables mvLXN[1] and refIdxLXN are further processed as follows.

- If the current macroblock is a field macroblock and the macroblock mbAddrN is a frame macroblock

$$\text{mvLXN}[1] = \text{mvLXN}[1] / 2 \quad (8-214)$$

$$\text{refIdxLXN} = \text{refIdxLXN} * 2 \quad (8-215)$$

- Otherwise, if the current macroblock is a frame macroblock and the macroblock mbAddrN is a field macroblock

$$\text{mvLXN}[1] = \text{mvLXN}[1] * 2 \quad (8-216)$$

$$\text{refIdxLXN} = \text{refIdxLXN} / 2 \quad (8-217)$$

- Otherwise, the vertical motion vector component $\text{mvLXN}[1]$ and the reference index refIdxLXN remain unchanged.

8.4.1.4 Derivation process for chroma motion vectors

This process is only invoked when chroma_format_idc is not equal to 0 (monochrome).

Inputs to this process are a luma motion vector mvLX and a reference index refIdxLX .

Output of this process is a chroma motion vector mvCLX .

A chroma motion vector is derived from the corresponding luma motion vector.

The precision of the chroma motion vector components is $1 \div (4 * \text{SubWidthC})$ horizontally and $1 \div (4 * \text{SubHeightC})$ vertically.

NOTE – For example, when using the 4:2:0 chroma format, since the units of luma motion vectors are one-quarter luma sample units and chroma has half horizontal and vertical resolution compared to luma, the units of chroma motion vectors are one-eighth chroma sample units, i.e., a value of 1 for the chroma motion vector refers to a one-eighth chroma sample displacement. For example, when the luma vector applies to 8x16 luma samples, the corresponding chroma vector in 4:2:0 chroma format applies to 4x8 chroma samples and when the luma vector applies to 4x4 luma samples, the corresponding chroma vector in 4:2:0 chroma format applies to 2x2 chroma samples.

For the derivation of the motion vector mvCLX , the following applies.

- If chroma_format_idc is not equal to 1 or the current macroblock is a frame macroblock, the horizontal and vertical components of the chroma motion vector mvCLX are derived as

$$\text{mvCLX}[0] = \text{mvLX}[0] \quad (8-218)$$

$$\text{mvCLX}[1] = \text{mvLX}[1] \quad (8-219)$$

- Otherwise (chroma_format_idc is equal to 1 and the current macroblock is a field macroblock), only the horizontal component of the chroma motion vector $\text{mvCLX}[0]$ is derived using Equation 8-218. The vertical component of the chroma motion vector $\text{mvCLX}[1]$ is dependent on the parity of the current field or the current macroblock and the reference picture, which is referred by the reference index refIdxLX . $\text{mvCLX}[1]$ is derived from $\text{mvLX}[1]$ according to Table 8-10.

Table 8-10 – Derivation of the vertical component of the chroma vector in field coding mode

Parity conditions		$\text{mvCLX}[1]$
Reference picture (refIdxLX)	Current field (picture/macroblock)	
Top field	Bottom field	$\text{mvLX}[1] + 2$
Bottom field	Top field	$\text{mvLX}[1] - 2$
Otherwise		$\text{mvLX}[1]$

8.4.2 Decoding process for Inter prediction samples

Inputs to this process are

- a macroblock partition mbPartIdx ,
- a sub-macroblock partition subMbPartIdx .
- variables specifying partition width and height for luma and chroma (if available), partWidth , partHeight , partWidthC (if available) and partHeightC (if available)
- luma motion vectors mvL0 and mvL1 and when chroma_format_idc is not equal to 0 (monochrome) chroma motion vectors mvCL0 and mvCL1
- reference indices refIdxL0 and refIdxL1

- prediction list utilization flags, predFlagL0 and predFlagL1

Outputs of this process are

- the Inter prediction samples predPart, which are a (partWidth)x(partHeight) array predPart_L of prediction luma samples, and when chroma_format_idc is not equal to 0 (monochrome) two (partWidthC)x(partHeightC) arrays predPart_{Cb}, predPart_{Cr} of prediction chroma samples, one for each of the chroma components Cb and Cr.

Let predPartL0_L and predPartL1_L be (partWidth)x(partHeight) arrays of predicted luma sample values and when chroma_format_idc is not equal to 0 (monochrome) predPartL0_{Cb}, predPartL1_{Cb}, predPartL0_{Cr}, and predPartL1_{Cr} be (partWidthC)x(partHeightC) arrays of predicted chroma sample values.

For LX being replaced by either L0 or L1 in the variables predFlagLX, RefPicListX, refIdxLX, refPicLX, predPartLX, the following is specified.

When predFlagLX is equal to 1, the following applies.

- The reference picture consisting of an ordered two-dimensional array refPicLX_L of luma samples and when chroma_format_idc is not equal to 0 (monochrome) two ordered two-dimensional arrays refPicLX_{Cb} and refPicLX_{Cr} of chroma samples is derived by invoking the process specified in subclause 8.4.2.1 with refIdxLX and RefPicListX given as input.
- The array predPartLX_L and when chroma_format_idc is not equal to 0 (monochrome) the arrays predPartLX_{Cb} and predPartLX_{Cr} are derived by invoking the process specified in subclause 8.4.2.2 with the current partition specified by mbPartIdx\subMbPartIdx, the motion vectors mvLX, mvCLX (if available), and the reference arrays with refPicLX_L, refPicLX_{Cb} (if available), and refPicLX_{Cr} (if available) given as input.

For C being replaced by L, Cb (if available), or Cr (if available), the array predPart_C of the prediction samples of component C is derived by invoking the process specified in subclause 8.4.2.3 with the current partition specified by mbPartIdx and subMbPartIdx and the array predPartL0_C and predPartL1_C as well as predFlagL0 and predFlagL1 given as input.

8.4.2.1 Reference picture selection process

Input to this process is a reference index refIdxLX.

Output of this process is a reference picture consisting of a two-dimensional array of luma samples refPicLX_L and two two-dimensional arrays of chroma samples refPicLX_{Cb} and refPicLX_{Cr}.

Depending on field_pic_flag, the reference picture list RefPicListX (which has been derived as specified in subclause 8.2.4) consists of the following.

- If field_pic_flag is equal to 1, each entry of RefPicListX is a reference field or a field of a reference frame.
- Otherwise (field_pic_flag is equal to 0), each entry of RefPicListX is a reference frame or a complementary reference field pair.

For the derivation of the reference picture, the following applies.

- If field_pic_flag is equal to 1, the reference field or field of a reference frame RefPicListX[refIdxLX] is the output. The output reference field or field of a reference frame consists of a (PicWidthInSamples_L)x(PicHeightInSamples_L) array of luma samples refPicLX_L and, when chroma_format_idc is not equal to 0 (monochrome), two (PicWidthInSamples_C)x(PicHeightInSamples_C) arrays of chroma samples refPicLX_{Cb} and refPicLX_{Cr}.
- Otherwise (field_pic_flag is equal to 0), the following applies.
 - If the current macroblock is a frame macroblock, the reference frame or complementary reference field pair RefPicListX[refIdxLX] is the output. The output reference frame or complementary reference field pair consists of a (PicWidthInSamples_L)x(PicHeightInSamples_L) array of luma samples refPicLX_L and, when chroma_format_idc is not equal to 0 (monochrome), two (PicWidthInSamples_C)x(PicHeightInSamples_C) arrays of chroma samples refPicLX_{Cb} and refPicLX_{Cr}.
 - Otherwise (the current macroblock is a field macroblock), the following applies.
 - Let refFrame be the reference frame or complementary reference field pair RefPicListX[refIdxLX / 2].
 - The field of refFrame is selected as follows.
 - If refIdxLX % 2 is equal to 0, the field of refFrame that has the same parity as the current macroblock is the output.

- Otherwise ($\text{refIdxLX} \% 2$ is equal to 1), the field of refFrame that has the opposite parity as the current macroblock is the output.
- The output reference field or field of a reference frame consists of a $(\text{PicWidthInSamples}_L) \times (\text{PicHeightInSamples}_L / 2)$ array of luma samples refPicLX_L and, when chroma_format_idc is not equal to 0 (monochrome), two $(\text{PicWidthInSamples}_C) \times (\text{PicHeightInSamples}_C / 2)$ arrays of chroma samples refPicLX_{Cb} and refPicLX_{Cr} .

The reference picture sample arrays refPicLX_L , refPicLX_{Cb} (if available), and refPicLX_{Cr} (if available) correspond to decoded sample arrays S_L , S_{Cb} (if available), S_{Cr} (if available) derived in subclause 8.7 for a previously-decoded reference field or reference frame or complementary reference field pair or field of a reference frame.

8.4.2.2 Fractional sample interpolation process

Inputs to this process are

- the current partition given by its partition index mbPartIdx and its sub-macroblock partition index subMbPartIdx ,
- the width and height partWidth , partHeight of this partition in luma-sample units,
- a luma motion vector mvLX given in quarter-luma-sample units,
- a chroma motion vector mvCLX given in eighth-chroma-sample units, and
- the selected reference picture sample arrays refPicLX_L , refPicLX_{Cb} , and refPicLX_{Cr}

Outputs of this process are

- a $(\text{partWidth}) \times (\text{partHeight})$ array predPartLX_L of prediction luma sample values and
- when chroma_format_idc is not equal to 0 (monochrome) two $(\text{partWidthC}) \times (\text{partHeightC})$ arrays predPartLX_{Cb} , and predPartLX_{Cr} of prediction chroma sample values.

Let (x_{A_L}, y_{A_L}) be the location given in full-sample units of the upper-left luma sample of the current partition given by $\text{mbPartIdx} \backslash \text{subMbPartIdx}$ relative to the upper-left luma sample location of the given two-dimensional array of luma samples.

Let (x_{Int_L}, y_{Int_L}) be a luma location given in full-sample units and (x_{Frac_L}, y_{Frac_L}) be an offset given in quarter-sample units. These variables are used only inside this subclause for specifying general fractional-sample locations inside the reference sample arrays refPicLX_L , refPicLX_{Cb} (if available), and refPicLX_{Cr} (if available).

For each luma sample location $(0 \leq x_L < \text{partWidth}, 0 \leq y_L < \text{partHeight})$ inside the prediction luma sample array predPartLX_L , the corresponding prediction luma sample value $\text{predPartLX}_L[x_L, y_L]$ is derived as follows:

- The variables x_{Int_L} , y_{Int_L} , x_{Frac_L} , and y_{Frac_L} are derived by

$$x_{Int_L} = x_{A_L} + (\text{mvLX}[0] \gg 2) + x_L \quad (8-220)$$

$$y_{Int_L} = y_{A_L} + (\text{mvLX}[1] \gg 2) + y_L \quad (8-221)$$

$$x_{Frac_L} = \text{mvLX}[0] \& 3 \quad (8-222)$$

$$y_{Frac_L} = \text{mvLX}[1] \& 3 \quad (8-223)$$

- The prediction luma sample value $\text{predPartLX}_L[x_L, y_L]$ is derived by invoking the process specified in subclause 8.4.2.2.1 with (x_{Int_L}, y_{Int_L}) , (x_{Frac_L}, y_{Frac_L}) and refPicLX_L given as input.

When chroma_format_idc is not equal to 0 (monochrome), the following applies.

Let (x_{Int_C}, y_{Int_C}) be a chroma location given in full-sample units and (x_{Frac_C}, y_{Frac_C}) be an offset given in one-eighth sample units. These variables are used only inside this subclause for specifying general fractional-sample locations inside the reference sample arrays refPicLX_{Cb} , and refPicLX_{Cr} .

For each chroma sample location $(0 \leq x_C < \text{partWidthC}, 0 \leq y_C < \text{partHeightC})$ inside the prediction chroma sample arrays predPartLX_{Cb} and predPartLX_{Cr} , the corresponding prediction chroma sample values $\text{predPartLX}_{Cb}[x_C, y_C]$ and $\text{predPartLX}_{Cr}[x_C, y_C]$ are derived as follows:

- Depending on chroma_format_idc , the variables x_{Int_C} , y_{Int_C} , x_{Frac_C} , and y_{Frac_C} are derived as follows.
 - If chroma_format_idc is equal to 1,

$$xInt_C = (xA_L / SubWidthC) + (mvCLX[0] >> 3) + x_C \quad (8-224)$$

$$yInt_C = (yA_L / SubHeightC) + (mvCLX[1] >> 3) + y_C \quad (8-225)$$

$$xFrac_C = mvCLX[0] \& 7 \quad (8-226)$$

$$yFrac_C = mvCLX[1] \& 7 \quad (8-227)$$

- Otherwise, if chroma_format_idc is equal to 2,

$$xInt_C = (xA_L / SubWidthC) + (mvCLX[0] >> 3) + x_C \quad (8-228)$$

$$yInt_C = (yA_L / SubHeightC) + (mvCLX[1] >> 2) + y_C \quad (8-229)$$

$$xFrac_C = mvCLX[0] \& 7 \quad (8-230)$$

$$yFrac_C = (mvCLX[1] \& 3) << 1 \quad (8-231)$$

- Otherwise (chroma_format_idc is equal to 3),

$$xInt_C = (xA_L / SubWidthC) + (mvCLX[0] >> 2) + x_C \quad (8-232)$$

$$yInt_C = (yA_L / SubHeightC) + (mvCLX[1] >> 2) + y_C \quad (8-233)$$

$$xFrac_C = (mvCLX[0] \& 3) << 1 \quad (8-234)$$

$$yFrac_C = (mvCLX[1] \& 3) << 1 \quad (8-235)$$

- The prediction sample value $predPartLX_{Cb}[x_C, y_C]$ is derived by invoking the process specified in subclause 8.4.2.2.2 with $(xInt_C, yInt_C)$, $(xFrac_C, yFrac_C)$ and $refPicLX_{Cb}$ given as input.
- The prediction sample value $predPartLX_{Cr}[x_C, y_C]$ is derived by invoking the process specified in subclause 8.4.2.2.2 with $(xInt_C, yInt_C)$, $(xFrac_C, yFrac_C)$ and $refPicLX_{Cr}$ given as input.

8.4.2.2.1 Luma sample interpolation process

Inputs to this process are

- a luma location in full-sample units $(xInt_L, yInt_L)$,
- a luma location offset in fractional-sample units $(xFrac_L, yFrac_L)$, and
- the luma sample array of the selected reference picture $refPicLX_L$

Output of this process is a predicted luma sample value $predPartLX_L[x_L, y_L]$.

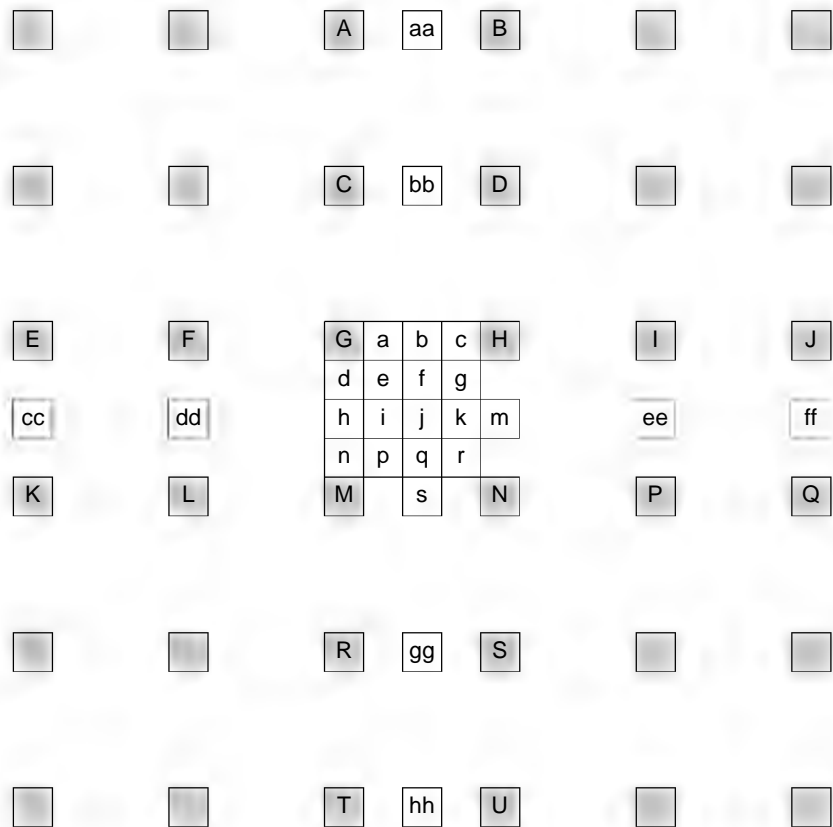


Figure 8-4 – Integer samples (shaded blocks with upper-case letters) and fractional sample positions (un-shaded blocks with lower-case letters) for quarter sample luma interpolation

The variable $\text{refPicHeightEffective}_L$, which is the height of the effective reference picture luma array, is derived as follows.

- If MbaffFrameFlag is equal to 0 or $\text{mb_field_decoding_flag}$ is equal to 0, $\text{refPicHeightEffective}_L$ is set equal to $\text{PicHeightInSamples}_L$.
- Otherwise (MbaffFrameFlag is equal to 1 and $\text{mb_field_decoding_flag}$ is equal to 1), $\text{refPicHeightEffective}_L$ is set equal to $\text{PicHeightInSamples}_L / 2$.

In Figure 8-4, the positions labelled with upper-case letters within shaded blocks represent luma samples at full-sample locations inside the given two-dimensional array refPicLX_L of luma samples. These samples may be used for generating the predicted luma sample value $\text{predPartLX}_L[x_L, y_L]$. The locations (xZ_L, yZ_L) for each of the corresponding luma samples Z , where Z may be A, B, C, D, E, F, G, H, I, J, K, L, M, N, P, Q, R, S, T, or U, inside the given array refPicLX_L of luma samples are derived as follows:

$$\begin{aligned} xZ_L &= \text{Clip3}(0, \text{PicWidthInSamples}_L - 1, x\text{Int}_L + xDZ_L) \\ yZ_L &= \text{Clip3}(0, \text{refPicHeightEffective}_L - 1, y\text{Int}_L + yDZ_L) \end{aligned} \quad (8-236)$$

Table 8-11 specifies (xDZ_L, yDZ_L) for different replacements of Z .

Table 8-11 – Differential full-sample luma locations

Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	P	Q	R	S	T	U
xDZ_L	0	1	0	1	-2	-1	0	1	2	3	-2	-1	0	1	2	3	0	1	0	1
yDZ_L	-2	-2	-1	-1	0	0	0	0	0	0	1	1	1	1	1	1	2	2	3	3

Given the luma samples ‘A’ to ‘U’ at full-sample locations (x_{A_L}, y_{A_L}) to (x_{U_L}, y_{U_L}), the luma samples ‘a’ to ‘s’ at fractional sample positions are derived by the following rules. The luma prediction values at half sample positions are derived by applying a 6-tap filter with tap values (1, -5, 20, 20, -5, 1). The luma prediction values at quarter sample positions are derived by averaging samples at full and half sample positions. The process for each fractional position is described below.

- The samples at half sample positions labelled b are derived by first calculating intermediate values denoted as b_1 by applying the 6-tap filter to the nearest integer position samples in the horizontal direction. The samples at half sample positions labelled h are derived by first calculating intermediate values denoted as h_1 by applying the 6-tap filter to the nearest integer position samples in the vertical direction:

$$b_1 = (E - 5 * F + 20 * G + 20 * H - 5 * I + J) \quad (8-237)$$

$$h_1 = (A - 5 * C + 20 * G + 20 * M - 5 * R + T) \quad (8-238)$$

The final prediction values b and h are derived using:

$$b = \text{Clip1}_Y((b_1 + 16) \gg 5) \quad (8-239)$$

$$h = \text{Clip1}_Y((h_1 + 16) \gg 5) \quad (8-240)$$

- The samples at half sample position labelled as j are derived by first calculating intermediate value denoted as j_1 by applying the 6-tap filter to the intermediate values of the closest half sample positions in either the horizontal or vertical direction because these yield an equal result.

$$j_1 = cc - 5 * dd + 20 * h_1 + 20 * m_1 - 5 * ee + ff, \text{ or} \quad (8-241)$$

$$j_1 = aa - 5 * bb + 20 * b_1 + 20 * s_1 - 5 * gg + hh \quad (8-242)$$

where intermediate values denoted as aa, bb, gg, s_1 and hh are derived by applying the 6-tap filter horizontally in the same manner as the derivation of b_1 and intermediate values denoted as cc, dd, ee, m_1 and ff are derived by applying the 6-tap filter vertically in the same manner as the derivation of h_1 . The final prediction value j are derived using:

$$j = \text{Clip1}_Y((j_1 + 512) \gg 10) \quad (8-243)$$

- The final prediction values s and m are derived from s_1 and m_1 in the same manner as the derivation of b and h, as given by:

$$s = \text{Clip1}_Y((s_1 + 16) \gg 5) \quad (8-244)$$

$$m = \text{Clip1}_Y((m_1 + 16) \gg 5) \quad (8-245)$$

- The samples at quarter sample positions labelled as a, c, d, n, f, i, k, and q are derived by averaging with upward rounding of the two nearest samples at integer and half sample positions using:

$$a = (G + b + 1) \gg 1 \quad (8-246)$$

$$c = (H + b + 1) \gg 1 \quad (8-247)$$

$$d = (G + h + 1) \gg 1 \quad (8-248)$$

$$n = (M + h + 1) \gg 1 \quad (8-249)$$

$$f = (b + j + 1) \gg 1 \quad (8-250)$$

$$i = (h + j + 1) \gg 1 \quad (8-251)$$

$$k = (j + m + 1) \gg 1 \quad (8-252)$$

$$q = (j + s + 1) \gg 1. \quad (8-253)$$

- The samples at quarter sample positions labelled as e, g, p, and r are derived by averaging with upward rounding of the two nearest samples at half sample positions in the diagonal direction using

$$e = (b + h + 1) \gg 1 \quad (8-254)$$

$$g = (b + m + 1) \gg 1 \quad (8-255)$$

$$p = (h + s + 1) \gg 1 \quad (8-256)$$

$$r = (m + s + 1) \gg 1. \quad (8-257)$$

The luma location offset in fractional-sample units ($x_{\text{Frac}_L}, y_{\text{Frac}_L}$) specifies which of the generated luma samples at full-sample and fractional-sample locations is assigned to the predicted luma sample value $\text{predPartLX}_L[x_L, y_L]$. This assignment is done according to Table 8-12. The value of $\text{predPartLX}_L[x_L, y_L]$ is the output.

Table 8-12 – Assignment of the luma prediction sample $\text{predPartLX}_L[x_L, y_L]$

$x\text{Frac}_L$	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
$y\text{Frac}_L$	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
$\text{predPartLX}_L[x_L, y_L]$	G	d	h	n	a	e	i	p	b	f	j	q	c	g	k	r

8.4.2.2.2 Chroma sample interpolation process

This process shall only be invoked when chroma_format_idc is not equal to 0 (monochrome).

Inputs to this process are

- a chroma location in full-sample units ($x\text{Int}_C, y\text{Int}_C$),
- a chroma location offset in fractional-sample units ($x\text{Frac}_C, y\text{Frac}_C$), and
- chroma component samples from the selected reference picture refPicLX_C .

Output of this process is a predicted chroma sample value $\text{predPartLX}_C[x_C, y_C]$.

In Figure 8-5, the positions labelled with A, B, C, and D represent chroma samples at full-sample locations inside the given two-dimensional array refPicLX_C of chroma samples.

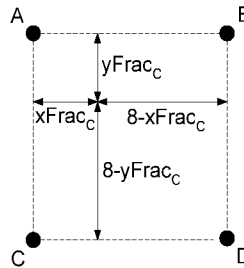


Figure 8-5 – Fractional sample position dependent variables in chroma interpolation and surrounding integer position samples A, B, C, and D

The variable $\text{refPicHeightEffective}_C$, which is the height of the effective reference picture chroma array, is derived as follows.

- If MbaffFrameFlag is equal to 0 or $\text{mb_field_decoding_flag}$ is equal to 0, $\text{refPicHeightEffective}_C$ is set equal to $\text{PicHeightInSamples}_C$.
- Otherwise (MbaffFrameFlag is equal to 1 and $\text{mb_field_decoding_flag}$ is equal to 1), $\text{refPicHeightEffective}_C$ is set equal to $\text{PicHeightInSamples}_C / 2$.

The sample coordinates specified in Equations 8-258 through 8-265 are used for generating the predicted chroma sample value $\text{predPartLX}_C[x_C, y_C]$.

$$x_{A_C} = \text{Clip3}(0, \text{PicWidthInSamples}_C - 1, x\text{Int}_C) \quad (8-258)$$

$$x_{B_C} = \text{Clip3}(0, \text{PicWidthInSamples}_C - 1, x\text{Int}_C + 1) \quad (8-259)$$

$$x_{C_C} = \text{Clip3}(0, \text{PicWidthInSamples}_C - 1, x\text{Int}_C) \quad (8-260)$$

$$x_{D_C} = \text{Clip3}(0, \text{PicWidthInSamples}_C - 1, x\text{Int}_C + 1) \quad (8-261)$$

$$y_{A_C} = \text{Clip3}(0, \text{refPicHeightEffective}_C - 1, y\text{Int}_C) \quad (8-262)$$

$$y_{B_C} = \text{Clip3}(0, \text{refPicHeightEffective}_C - 1, y\text{Int}_C) \quad (8-263)$$

$$y_{C_C} = \text{Clip3}(0, \text{refPicHeightEffective}_C - 1, y\text{Int}_C + 1) \quad (8-264)$$

$$y_{D_C} = \text{Clip3}(0, \text{refPicHeightEffective}_C - 1, y\text{Int}_C + 1) \quad (8-265)$$

Given the chroma samples A, B, C, and D at full-sample locations specified in Equations 8-258 through 8-265, the predicted chroma sample value $\text{predPartLXC}[x_C, y_C]$ is derived as follows:

$$\text{predPartLXC}[x_C, y_C] = ((8 - x_{\text{Frac}_C}) * (8 - y_{\text{Frac}_C}) * A + x_{\text{Frac}_C} * (8 - y_{\text{Frac}_C}) * B + (8 - x_{\text{Frac}_C}) * y_{\text{Frac}_C} * C + x_{\text{Frac}_C} * y_{\text{Frac}_C} * D + 32) \gg 6 \quad (8-266)$$

8.4.2.3 Weighted sample prediction process

Inputs to this process are

- mbPartIdx : the current partition given by the partition index
- subMbPartIdx : the sub-macroblock partition index
- predFlagL0 and predFlagL1 : prediction list utilization flags
- predPartLXL : a $(\text{partWidth}) \times (\text{partHeight})$ array of prediction luma samples (with LX being replaced by L0 or L1 depending on predFlagL0 and predFlagL1)
- when chroma_format_idc is not equal to 0 (monochrome), predPartLXC_b and predPartLXC_r : $(\text{partWidthC}) \times (\text{partHeightC})$ arrays of prediction chroma samples, one for each of the chroma components Cb and Cr (with LX being replaced by L0 or L1 depending on predFlagL0 and predFlagL1)

Outputs of this process are

- predPart_L : a $(\text{partWidth}) \times (\text{partHeight})$ array of prediction luma samples and
- when chroma_format_idc is not equal to 0 (monochrome), predPart_{Cb} , and predPart_{Cr} : $(\text{partWidthC}) \times (\text{partHeightC})$ arrays of prediction chroma samples, one for each of the chroma components Cb and Cr.

For macroblocks or partitions with predFlagL0 equal to 1 in P and SP slices, the following applies.

- If $\text{weighted_pred_flag}$ is equal to 0, the default weighted sample prediction process as described in subclause 8.4.2.3.1 is invoked with the same inputs and outputs as the process described in this subclause.
- Otherwise ($\text{weighted_pred_flag}$ is equal to 1), the explicit weighted prediction process as described in subclause 8.4.2.3.2 is invoked with the same inputs and outputs as the process described in this subclause.

For macroblocks or partitions with predFlagL0 or predFlagL1 equal to 1 in B slices, the following applies.

- If $\text{weighted_bipred_idc}$ is equal to 0, the default weighted sample prediction process as described in subclause 8.4.2.3.1 is invoked with the same inputs and outputs as the process described in this subclause.
- Otherwise, if $\text{weighted_bipred_idc}$ is equal to 1, the explicit weighted sample prediction process as described in subclause 8.4.2.3.2, for macroblocks or partitions with predFlagL0 or predFlagL1 equal to 1 with the same inputs and outputs as the process described in this subclause.
- Otherwise ($\text{weighted_bipred_idc}$ is equal to 2), the following applies.
 - If predFlagL0 is equal to 1 and predFlagL1 is equal to 1, the implicit weighted sample prediction as described in subclause 8.4.2.3.2 is invoked with the same inputs and outputs as the process described in this subclause.
 - Otherwise (predFlagL0 or predFlagL1 are equal to 1 but not both), the default weighted sample prediction process as described in subclause 8.4.2.3.1 is invoked with the same inputs and outputs as the process described in this subclause.

8.4.2.3.1 Default weighted sample prediction process

Input to this process are the same as specified in subclause 8.4.2.3.

Output of this process are the same as specified in subclause 8.4.2.3.

Depending on the available component for which the prediction block is derived, the following applies.

- If the luma sample prediction values $\text{predPart}_L[x, y]$ are derived, the following applies with C set equal to L, x set equal to 0 .. $\text{partWidth} - 1$, and y set equal to 0 .. $\text{partHeight} - 1$.
- Otherwise, if the chroma Cb component sample prediction values $\text{predPart}_{Cb}[x, y]$ are derived, the following applies with C set equal to Cb, x set equal to 0 .. $\text{partWidthC} - 1$, and y set equal to 0 .. $\text{partHeightC} - 1$.
- Otherwise (the chroma Cr component sample prediction values $\text{predPart}_{Cr}[x, y]$ are derived), the following applies with C set equal to Cr, x set equal to 0 .. $\text{partWidthC} - 1$, and y set equal to 0 .. $\text{partHeightC} - 1$.

The prediction sample values are derived as follows.

- If predFlagL0 is equal to 1 and predFlagL1 is equal to 0 for the current partition

$$\text{predPartC}[x, y] = \text{predPartL0C}[x, y] \quad (8-267)$$

- Otherwise, if predFlagL0 is equal to 0 and predFlagL1 is equal to 1 for the current partition

$$\text{predPartC}[x, y] = \text{predPartL1C}[x, y] \quad (8-268)$$

- Otherwise (predFlagL0 and predFlagL1 are equal to 1 for the current partition),

$$\text{predPartC}[x, y] = (\text{predPartL0C}[x, y] + \text{predPartL1C}[x, y] + 1) \gg 1. \quad (8-269)$$

8.4.2.3.2 Weighted sample prediction process

Input to this process are the same as specified in subclause 8.4.2.3.

Output of this process are the same as specified in subclause 8.4.2.3.

Depending on the available component for which the prediction block is derived, the following applies.

- If the luma sample prediction values $\text{predPartL}[x, y]$ are derived, the following applies with C set equal to L, x set equal to 0 .. partWidth - 1, and y set equal to 0 .. partHeight - 1.
- Otherwise, if the chroma Cb component sample prediction values $\text{predPartCb}[x, y]$ are derived, the following applies with C set equal to Cb, x set equal to 0 .. partWidthC - 1, and y set equal to 0 .. partHeightC - 1.
- Otherwise (the chroma Cr component sample prediction values $\text{predPartCr}[x, y]$ are derived), the following applies with C set equal to Cr, x set equal to 0 .. partWidthC - 1, and y set equal to 0 .. partHeightC - 1.

The prediction sample values are derived as follows

- If the partition $\text{mbPartIdx} \backslash \text{subMbPartIdx}$ has predFlagL0 equal to 1 and predFlagL1 equal to 0, the final predicted sample values $\text{predPartC}[x, y]$ are derived by

$$\begin{aligned} &\text{if}(\log\text{WD} \geq 1) \\ &\quad \text{predPartC}[x, y] = \text{Clip1C}((\text{predPartL0C}[x, y] * w_0 + 2^{\log\text{WD} - 1}) \gg \log\text{WD}) + o_0) \\ &\text{else} \\ &\quad \text{predPartC}[x, y] = \text{Clip1C}(\text{predPartL0C}[x, y] * w_0 + o_0) \end{aligned} \quad (8-270)$$

- Otherwise, if the partition $\text{mbPartIdx} \backslash \text{subMbPartIdx}$ has predFlagL0 equal to 0 and predFlagL1 equal to 1, the final predicted sample values $\text{predPartC}[x, y]$ are derived by

$$\begin{aligned} &\text{if}(\log\text{WD} \geq 1) \\ &\quad \text{predPartC}[x, y] = \text{Clip1C}((\text{predPartL1C}[x, y] * w_1 + 2^{\log\text{WD} - 1}) \gg \log\text{WD}) + o_1) \\ &\text{else} \\ &\quad \text{predPartC}[x, y] = \text{Clip1C}(\text{predPartL1C}[x, y] * w_1 + o_1) \end{aligned} \quad (8-271)$$

- Otherwise (the partition $\text{mbPartIdx} \backslash \text{subMbPartIdx}$ has both predFlagL0 and predFlagL1 equal to 1), the final predicted sample values $\text{predPartC}[x, y]$ are derived by

$$\text{predPartC}[x, y] = \text{Clip1C}((\text{predPartL0C}[x, y] * w_0 + \text{predPartL1C}[x, y] * w_1 + 2^{\log\text{WD}}) \gg (\log\text{WD} + 1)) + ((o_0 + o_1 + 1) \gg 1) \quad (8-272)$$

The variables in the above derivation for the prediction samples are derived as follows.

- If weighted_bipred_idc is equal to 2 and the slice_type is equal to B, implicit mode weighted prediction is used as follows.

$$\log\text{WD} = 5 \quad (8-273)$$

$$o_0 = 0 \quad (8-274)$$

$$o_1 = 0 \quad (8-275)$$

and w_0 and w_1 are derived as follows.

- The variables currPicOrField, pic0, and pic1 are derived as follows:
 - If field_pic_flag is equal to 0 and the current macroblock is a field macroblock, the following applies.
 - currPicOrField is the field of the current picture CurrPic that has the same parity as the current macroblock.
 - The variable pic0 is derived as follows.
 - If refIdxL0 % 2 is equal to 0, pic0 is the field of RefPicList0[refIdxL0 / 2] that has the same parity as the current macroblock.
 - Otherwise (refIdxL0 % 2 is not equal to 0), pic0 is the field of RefPicList0[refIdxL0 / 2] that has the opposite parity of the current macroblock.
 - The variable pic1 is derived as follows.
 - If refIdxL1 % 2 is equal to 0, pic1 is the field of RefPicList1[refIdxL1 / 2] that has the same parity as the current macroblock.
 - Otherwise (refIdxL1 % 2 is not equal to 0), pic1 is the field of RefPicList1[refIdxL1 / 2] that has the opposite parity of the current macroblock.
 - Otherwise (field_pic_flag is equal to 1 or the current macroblock is a frame macroblock), currPicOrField is the current picture CurrPic, pic1 is RefPicList1[refIdxL1], and pic0 is RefPicList0[refIdxL0].
- The variables tb, td, tx, and DistScaleFactor are derived from the values of currPicOrField, pic0, pic1 using Equations 8-198, 8-199, 8-194, and 8-195, respectively.
- If DiffPicOrderCnt(pic1, pic0) is equal to 0 or one or both of pic1 and pic0 is marked as "used for long-term reference" or (DistScaleFactor >> 2) < -64 or (DistScaleFactor >> 2) > 128, w_0 and w_1 are derived as

$$w_0 = 32 \quad (8-276)$$

$$w_1 = 32 \quad (8-277)$$

- Otherwise,

$$w_0 = 64 - (\text{DistScaleFactor} \gg 2) \quad (8-278)$$

$$w_1 = \text{DistScaleFactor} \gg 2 \quad (8-279)$$

- Otherwise (weighted_pred_flag is equal to 1 in P or SP slices or weighted_bipred_idc equal to 1 in B slices), explicit mode weighted prediction is used as follows.

- The variables refIdxL0WP and refIdxL1WP are derived as follows.
 - If MbaffFrameFlag is equal to 1 and the current macroblock is a field macroblock

$$\text{refIdxL0WP} = \text{refIdxL0} \gg 1 \quad (8-280)$$

$$\text{refIdxL1WP} = \text{refIdxL1} \gg 1 \quad (8-281)$$

- Otherwise (MbaffFrameFlag is equal to 0 or the current macroblock is a frame macroblock),

$$\text{refIdxL0WP} = \text{refIdxL0} \quad (8-282)$$

$$\text{refIdxL1WP} = \text{refIdxL1} \quad (8-283)$$

– The variables $\log\text{WD}$, w_0 , w_1 , o_0 , and o_1 are derived as follows.

– If C in $\text{predPart}_C[x, y]$ is replaced by L for luma samples

$$\log\text{WD} = \text{luma_log2_weight_denom} \quad (8-284)$$

$$w_0 = \text{luma_weight_l0}[\text{refIdxL0WP}] \quad (8-285)$$

$$w_1 = \text{luma_weight_l1}[\text{refIdxL1WP}] \quad (8-286)$$

$$o_0 = \text{luma_offset_l0}[\text{refIdxL0WP}] * (1 \ll (\text{BitDepth}_Y - 8)) \quad (8-287)$$

$$o_1 = \text{luma_offset_l1}[\text{refIdxL1WP}] * (1 \ll (\text{BitDepth}_Y - 8)) \quad (8-288)$$

– Otherwise (C in $\text{predPart}_C[x, y]$ is replaced by C_b or C_r for chroma samples, with $iCbCr = 0$ for C_b , $iCbCr = 1$ for C_r),

$$\log\text{WD} = \text{chroma_log2_weight_denom} \quad (8-289)$$

$$w_0 = \text{chroma_weight_l0}[\text{refIdxL0WP}][iCbCr] \quad (8-290)$$

$$w_1 = \text{chroma_weight_l1}[\text{refIdxL1WP}][iCbCr] \quad (8-291)$$

$$o_0 = \text{chroma_offset_l0}[\text{refIdxL0WP}][iCbCr] * (1 \ll (\text{BitDepth}_C - 8)) \quad (8-292)$$

$$o_1 = \text{chroma_offset_l1}[\text{refIdxL1WP}][iCbCr] * (1 \ll (\text{BitDepth}_C - 8)) \quad (8-293)$$

When explicit mode weighted prediction is used and the partition $\text{mbPartIdx} \backslash \text{subMbPartIdx}$ has both predFlagL0 and predFlagL1 equal to 1, the following constraint shall be obeyed

$$-128 \leq w_0 + w_1 \leq ((\log\text{WD} == 7) ? 127 : 128) \quad (8-294)$$

NOTE – For implicit mode weighted prediction, weights w_0 and w_1 are each guaranteed to be in the range of -64..128 and the constraint expressed in Equation 8-294, although not explicitly imposed, will always be met. For explicit mode weighted prediction with $\log\text{WD}$ equal to 7, when one of the two weights w_0 or w_1 is inferred to be equal to 128 (as a consequence of $\text{luma_weight_l0_flag}$, $\text{luma_weight_l1_flag}$, $\text{chroma_weight_l0_flag}$, or $\text{chroma_weight_l1_flag}$ equal to 0), the other weight (w_1 or w_0) must have a negative value in order for the constraint expressed in Equation 8-294 to hold (and therefore the other flag $\text{luma_weight_l0_flag}$, $\text{luma_weight_l1_flag}$, $\text{chroma_weight_l0_flag}$, or $\text{chroma_weight_l1_flag}$ must be equal to 1).

8.5 Transform coefficient decoding process and picture construction process prior to deblocking filter process

Inputs to this process are Intra16x16DCLevel (if available), Intra16x16ACLevel (if available), LumaLevel (if available), LumaLevel8x8 (if available), ChromaDCLevel (if available), ChromaACLevel (if available), and available Inter or Intra prediction sample arrays for the current macroblock for the applicable components pred_L , pred_{C_b} , or pred_{C_r} .

NOTE 1 – When decoding a macroblock in Intra_4x4 (or Intra_8x8) prediction mode, the luma component of the macroblock prediction array may not be complete, since for each $4x4$ (or $8x8$) luma block, the Intra_4x4 (or Intra_8x8) prediction process for luma samples as specified in subclause 8.3.1 (or 8.3.2) and the process specified in this subclause are iterated.

Outputs of this process are the constructed sample arrays prior to the deblocking filter process for the applicable components S'_L , S'_{C_b} , or S'_{C_r} .

NOTE 2 – When decoding a macroblock in Intra_4x4 (or Intra_8x8) prediction mode, the luma component of the macroblock constructed sample arrays prior to the deblocking filter process may not be complete, since for each $4x4$ (or $8x8$) luma block, the

Intra_4x4 (or Intra_8x8) prediction process for luma samples as specified in subclause 8.3.1 (or 8.3.2) and the process specified in this subclause are iterated.

This subclause specifies transform coefficient decoding and picture construction prior to the deblocking filter process.

When the current macroblock is coded as P_Skip or B_Skip, all values of LumaLevel, LumaLevel8x8, ChromaDCLevel, ChromaACLevel are set equal to 0 for the current macroblock.

When residual_colour_transform_flag is equal to 1, the residual colour transform process as specified in subclause 8.5.13 is invoked.

8.5.1 Specification of transform decoding process for 4x4 luma residual blocks

This specification applies when transform_size_8x8_flag is equal to 0.

When the current macroblock prediction mode is not equal to Intra_16x16, the variable LumaLevel contains the levels for the luma transform coefficients. For a 4x4 luma block indexed by luma4x4BlkIdx = 0..15, the following ordered steps are specified.

1. The inverse transform coefficient scanning process as described in subclause 8.5.5 is invoked with LumaLevel[luma4x4BlkIdx] as the input and the two-dimensional array c as the output.
2. The scaling and transformation process for residual 4x4 blocks as specified in subclause 8.5.10 is invoked with c as the input and r as the output.
3. When residual_colour_transform_flag is equal to 1, the variable $R_{Y,ij}$ is set equal to r_{ij} with $i, j = 0..3$ and this process is suspended until after completion of the residual colour transform process as specified in subclause 8.5.13, after the completion of which, the variable r_{ij} is set equal to $R_{G,ij}$ with $i, j = 0..3$ and this process is continued.
4. The position of the upper-left sample of a 4x4 luma block with index luma4x4BlkIdx inside the macroblock is derived by invoking the inverse 4x4 luma block scanning process in subclause 6.4.3 with luma4x4BlkIdx as the input and the output being assigned to (xO, yO).
5. The 4x4 array u with elements u_{ij} for $i, j = 0..3$ is derived as

$$u_{ij} = \text{Clip1}_Y(\text{pred}_L[xO + j, yO + i] + r_{ij}) \quad (8-295)$$

When qpprime_y_zero_transform_bypass_flag is equal to 1 and QP'_Y is equal to 0, the bitstream shall not contain data that result in a value of u_{ij} as computed by Equation 8-295 that is not equal to $\text{pred}_L[xO + j, yO + i] + r_{ij}$.

6. The picture construction process prior to deblocking filter process in subclause 8.5.12 is invoked with luma4x4BlkIdx and u as the inputs.

8.5.2 Specification of transform decoding process for luma samples of Intra_16x16 macroblock prediction mode

When the current macroblock prediction mode is equal to Intra_16x16, the variables Intra16x16DCLevel and Intra16x16ACLevel contain the levels for the luma transform coefficients. The transform coefficient decoding proceeds in the following ordered steps:

1. The 4x4 luma DC transform coefficients of all 4x4 luma blocks of the macroblock are decoded.
 - a. The inverse transform coefficient scanning process as described in subclause 8.5.5 is invoked with Intra16x16DCLevel as the input and the two-dimensional array c as the output.
 - b. The scaling and transformation process for luma DC transform coefficients for Intra_16x16 macroblock type as specified in subclause 8.5.8 is invoked with c as the input and dcY as the output.
2. For a 4x4 luma block indexed by luma4x4BlkIdx = 0..15, the following ordered steps are specified.
 - a. The variable lumaList, which is a list of 16 entries, is derived. The first entry of lumaList is the corresponding value from the array dcY. Figure 8-6 shows the assignment of the indices of the array dcY to the luma4x4BlkIdx. The two numbers in the small squares refer to indices i and j in dcY_{ij} , and the numbers in large squares refer to luma4x4BlkIdx.

00 0	01 1	02 4	03 5
10 2	11 3	12 6	13 7
20 8	21 9	22 12	23 13
30 10	31 11	32 14	33 15

Figure 8-6 – Assignment of the indices of dcY to luma4x4BlkIdx

The elements in lumaList with index $k = 1..15$ are specified as

$$\text{lumaList}[k] = \text{Intra16x16ACLevel}[\text{luma4x4BlkIdx}][k - 1] \quad (8-296)$$

- b. The inverse transform coefficient scanning process as described in subclause 8.5.5 is invoked with lumaList as the input and the two-dimensional array c as the output.
- c. The scaling and transformation process for residual 4x4 blocks as specified in subclause 8.5.10 is invoked with c as the input and r as the output.
- d. When residual_colour_transform_flag is equal to 1, the variable $R_{Y,ij}$ is set equal to r_{ij} with $i, j = 0..3$ and this process is suspended until after completion of the residual colour transform process as specified in subclause 8.5.13, after the completion of which, the variable r_{ij} is set equal to $R_{G,ij}$ with $i, j = 0..3$ and this process is continued.
- e. The position of the upper-left sample of a 4x4 luma block with index luma4x4BlkIdx inside the macroblock is derived by invoking the inverse 4x4 luma block scanning process in subclause 6.4.3 with luma4x4BlkIdx as the input and the output being assigned to (xO, yO).
- f. The 4x4 array u with elements u_{ij} for $i, j = 0..3$ is derived as

$$u_{ij} = \text{Clip1}_Y(\text{pred}_L[xO + j, yO + i] + r_{ij}) \quad (8-297)$$

When qpprime_y_zero_transform_bypass_flag is equal to 1 and QP'_Y is equal to 0, the bitstream shall not contain data that result in a value of u_{ij} as computed by 8-297 that is not equal to $\text{pred}_L[xO + j, yO + i] + r_{ij}$.

- g. The picture construction process prior to deblocking filter process in subclause 8.5.12 is invoked with luma4x4BlkIdx and u as the inputs.

8.5.3 Specification of transform decoding process for 8x8 luma residual blocks

This specification applies when transform_size_8x8_flag is equal to 1.

The variable LumaLevel8x8[luma8x8BlkIdx] with luma8x8BlkIdx = 0..3 contains the levels for the luma transform coefficients for the luma 8x8 block with index luma8x8BlkIdx.

For an 8x8 luma block indexed by luma8x8BlkIdx = 0..3, the following ordered steps are specified.

1. The inverse scanning process for 8x8 luma transform coefficients as described in subclause 8.5.6 is invoked with LumaLevel8x8[luma8x8BlkIdx] as the input and the two-dimensional array c as the output.
2. The scaling and transformation process for residual 8x8 blocks as specified in subclause 8.5.11 is invoked with c as the input and r as the output.
3. When residual_colour_transform_flag is equal to 1, the variable $R_{Y,ij}$ is set equal to r_{ij} with $i, j = 0..7$ and this process is suspended until after completion of the residual colour transform process as specified in subclause 8.5.13, after the completion of which, the variable r_{ij} is set equal to $R_{G,ij}$ with $i, j = 0..7$ and this process is continued.

4. The position of the upper-left sample of an 8x8 luma block with index luma8x8BlkIdx inside the macroblock is derived by invoking the inverse 8x8 luma block scanning process in subclause 6.4.4 with luma8x8BlkIdx as the input and the output being assigned to (xO, yO).
5. The 8x8 array u with elements u_{ij} for $i, j = 0..7$ is derived as

$$u_{ij} = \text{Clip1}_Y(\text{pred}_L[xO + j, yO + i] + r_{ij}) \quad (8-298)$$

When `qpprime_y_zero_transform_bypass_flag` is equal to 1 and QP'_Y is equal to 0, the bitstream shall not contain data that result in a value of u_{ij} as computed by Equation 8-298 that is not equal to $\text{pred}_L[xO + j, yO + i] + r_{ij}$.

6. The picture construction process prior to deblocking filter process in subclause 8.5.12 is invoked with luma8x8BlkIdx and u as the inputs.

8.5.4 Specification of transform decoding process for chroma samples

This process is invoked for each chroma component Cb and Cr separately.

For each chroma component, the variables `ChromaDCLevel[iCbCr]` and `ChromaACLevel[iCbCr]`, with `iCbCr` set equal to 0 for Cb and `iCbCr` set equal to 1 for Cr, contain the levels for both components of the chroma transform coefficients.

Let the variable `numChroma4x4Blks` be set equal to $(\text{MbWidthC} / 4) * (\text{MbHeightC} / 4)$.

For each chroma component, the transform decoding proceeds separately in the following ordered steps:

1. The `numChroma4x4Blks` chroma DC transform coefficients of the 4x4 chroma blocks of the component indexed by `iCbCr` of the macroblock are decoded.

- a. Depending on the variable `chroma_format_idc`, the following applies.
 - If `chroma_format_idc` is equal to 1, the 2x2 array c is derived using the inverse raster scanning process applied to `ChromaDCLevel` as follows

$$c = \begin{bmatrix} \text{ChromaDCLevel}[iCbCr][0] & \text{ChromaDCLevel}[iCbCr][1] \\ \text{ChromaDCLevel}[iCbCr][2] & \text{ChromaDCLevel}[iCbCr][3] \end{bmatrix} \quad (8-299)$$

- Otherwise, if `chroma_format_idc` is equal to 2, the 2x4 array c is derived using the inverse raster scanning process applied to `ChromaDCLevel` as follows

$$c = \begin{bmatrix} \text{ChromaDCLevel}[iCbCr][0] & \text{ChromaDCLevel}[iCbCr][2] \\ \text{ChromaDCLevel}[iCbCr][1] & \text{ChromaDCLevel}[iCbCr][5] \\ \text{ChromaDCLevel}[iCbCr][3] & \text{ChromaDCLevel}[iCbCr][6] \\ \text{ChromaDCLevel}[iCbCr][4] & \text{ChromaDCLevel}[iCbCr][7] \end{bmatrix} \quad (8-300)$$

- Otherwise (`chroma_format_idc` is equal to 3), the inverse scanning process for transform coefficients as specified in subclause 8.5.5 is invoked with `ChromaDCLevel[iCbCr]` as the input and the two-dimensional 4x4 array c as the output.
- b. The scaling and transformation process for chroma DC transform coefficients as specified in subclause 8.5.9 is invoked with c as the input and `dcC` as the output.
2. For each 4x4 chroma block indexed by `chroma4x4BlkIdx = 0..numChroma4x4Blks – 1` of the component indexed by `iCbCr`, the following ordered steps are specified.
 - a. The variable `chromaList`, which is a list of 16 entries, is derived. The first entry of `chromaList` is the corresponding value from the array `dcC`. Figure 8-7 shows the assignment of the indices of the array `dcC` to the `chroma4x4BlkIdx`. The two numbers in the small squares refer to indices i and j in `dcCij`, and the numbers in large squares refer to `chroma4x4BlkIdx`.

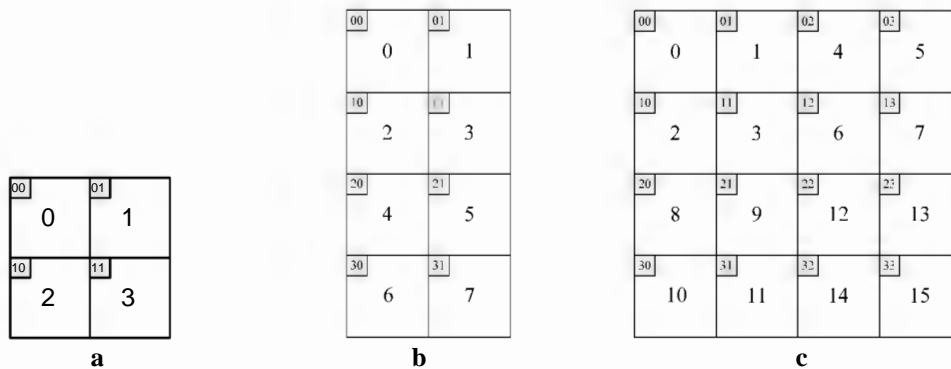


Figure 8-7 – Assignment of the indices of dcC to chroma4x4BlkIdx:

(a) chroma_format_idc equal to 1, (b) chroma_format_idc equal to 2, (c) chroma_format_idc equal to 3

The elements in chromaList with index $k = 1..15$ are specified as

$$\text{chromaList}[k] = \text{ChromaACLevel}[\text{chroma4x4BlkIdx}][k - 1] \quad (8-301)$$

- b. The inverse scanning process for transform coefficients as specified in subclause 8.5.9 is invoked with chromaList as the input and the two-dimensional array c as the output.
- c. The scaling and transformation process for residual 4x4 blocks as specified in subclause 8.5.10 is invoked with c as the input and r as the output.
- d. Depending on the variable chroma_format_idc, the position of the upper-left sample of a 4x4 chroma block with index chroma4x4BlkIdx inside the macroblock is derived as follows.

- If chroma_format_idc is equal to 1 or 2, the following applies.

$$xO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 0) \quad (8-302)$$

$$yO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 1) \quad (8-303)$$

- Otherwise (chroma_format_idc is equal to 3), the following applies.

$$xO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx} / 4, 8, 8, 16, 0) + \text{InverseRasterScan}(\text{chroma4x4BlkIdx} \% 4, 4, 4, 8, 0) \quad (8-304)$$

$$yO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx} / 4, 8, 8, 16, 1) + \text{InverseRasterScan}(\text{chroma4x4BlkIdx} \% 4, 4, 4, 8, 1) \quad (8-305)$$

- e. When residual_colour_transform_flag is equal to 1, the variable xO' is set equal to $xO \% (4 \ll \text{transform_size_8x8_flag})$, the variable yO' is set equal to $yO \% (4 \ll \text{transform_size_8x8_flag})$, and the following applies.

- If this process is invoked for the chroma component Cb, the variable $R_{Cb,mn}$ is set equal to r_{ij} with $i, j = 0..3$, $m = xO' + i$, $n = yO' + j$, and this process is suspended until after completion of the residual colour transform process as specified in subclause 8.5.13, after which the variable r_{ij} is set equal to $R_{B,mn}$ with $i, j = 0..3$, $m = xO' + i$, $n = yO' + j$ and this process is continued.

- Otherwise (this process is invoked for the chroma component Cr), the variable $R_{Cr,mn}$ is set equal to r_{ij} with $i, j = 0..3$, $m = xO' + i$, $n = yO' + j$ and this process is suspended until after the completion of the residual colour transform process as specified in subclause 8.5.13, after which the variable r_{ij} is set equal to $R_{R,mn}$ with $i, j = 0..3$, $m = xO' + i$, $n = yO' + j$ and this process is continued.

- f. The 4x4 array u with elements u_{ij} for $i, j = 0..3$ is derived as

$$u_{ij} = \text{Clip1}_c(\text{pred}_c[xO + j, yO + i] + r_{ij}) \quad (8-306)$$

When `qpprime_y_zero_transform_bypass_flag` is equal to 1 and QP'_Y is equal to 0, the bitstream shall not contain data that result in a value of u_{ij} as computed by Equation 8-306 that is not equal to $\text{pred}_c[xO + j, yO + i] + r_{ij}$.

- g. The picture construction process prior to deblocking filter process in subclause 8.5.12 is invoked with `chroma4x4BlkIdx` and u as the inputs.

8.5.5 Inverse scanning process for transform coefficients

Input to this process is a list of 16 values.

Output of this process is a variable c containing a two-dimensional array of 4x4 values. In the case of transform coefficients, these 4x4 values represent levels assigned to locations in the transform block. In the case of applying the inverse scanning process to a scaling list, the output variable c contains a two-dimensional array representing a 4x4 scaling matrix.

The inverse scanning process for transform coefficients maps the sequence of transform coefficient levels to the transform coefficient level positions. Table 8-13 specifies the two mappings: inverse zig-zag scan and inverse field scan. The inverse zig-zag scan is used for transform coefficients in frame macroblocks and the inverse field scan is used for transform coefficients in field macroblocks.

The inverse scanning process for scaling lists maps the sequence of scaling list entries to the positions in the corresponding scaling matrix. For this mapping, the inverse zig-zag scan is used.

Figure 8-8 illustrates the scans.

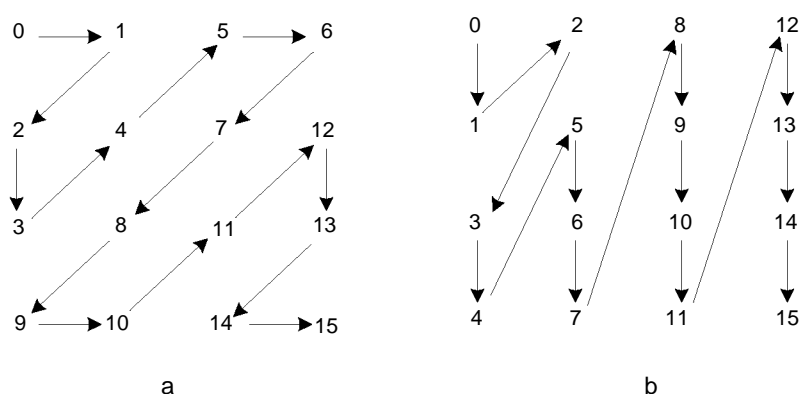


Figure 8-8 – 4x4 block scans. (a) Zig-zag scan. (b) Field scan (informative)

Table 8-13 provides the mapping from the index idx of input list of 16 elements to indices i and j of the two-dimensional array c .

Table 8-13 – Specification of mapping of idx to c_{ij} for zig-zag and field scan

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
zig-zag	c_{00}	c_{01}	c_{10}	c_{20}	c_{11}	c_{02}	c_{03}	c_{12}	c_{21}	c_{30}	c_{31}	c_{22}	c_{13}	c_{23}	c_{32}	c_{33}
field	c_{00}	c_{10}	c_{01}	c_{20}	c_{30}	c_{11}	c_{21}	c_{31}	c_{02}	c_{12}	c_{22}	c_{32}	c_{03}	c_{13}	c_{23}	c_{33}

8.5.6 Inverse scanning process for 8x8 luma transform coefficients

Input to this process is a list of 64 values.

Table 8-14 – Specification of mapping of idx to c_{ij} for 8x8 zig-zag and 8x8 field scan

idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
zig-zag	c ₀₀	c ₀₁	c ₁₀	c ₂₀	c ₁₁	c ₀₂	c ₀₃	c ₁₂	c ₂₁	c ₃₀	c ₄₀	c ₃₁	c ₂₂	c ₁₃	c ₀₄	c ₀₅
field	c ₀₀	c ₁₀	c ₂₀	c ₀₁	c ₁₁	c ₃₀	c ₄₀	c ₂₁	c ₀₂	c ₃₁	c ₅₀	c ₆₀	c ₇₀	c ₄₁	c ₁₂	c ₀₃

Table 8-14 (continued) – Specification of mapping of idx to c_{ij} for 8x8 zig-zag and 8x8 field scan

idx	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
zig-zag	c ₁₄	c ₂₃	c ₃₂	c ₄₁	c ₅₀	c ₆₀	c ₅₁	c ₄₂	c ₃₃	c ₂₄	c ₁₅	c ₀₆	c ₀₇	c ₁₆	c ₂₅	c ₃₄
field	c ₂₂	c ₅₁	c ₆₁	c ₇₁	c ₃₂	c ₁₃	c ₀₄	c ₂₃	c ₄₂	c ₅₂	c ₆₂	c ₇₂	c ₃₃	c ₁₄	c ₀₅	c ₂₄

Table 8-14 (continued) – Specification of mapping of idx to c_{ij} for 8x8 zig-zag and 8x8 field scan

idx	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
zig-zag	c ₄₃	c ₅₂	c ₆₁	c ₇₀	c ₇₁	c ₆₂	c ₅₃	c ₄₄	c ₃₅	c ₂₆	c ₁₇	c ₂₇	c ₃₆	c ₄₅	c ₅₄	c ₆₃
field	c ₄₃	c ₅₃	c ₆₃	c ₇₃	c ₃₄	c ₁₅	c ₀₆	c ₂₅	c ₄₄	c ₅₄	c ₆₄	c ₇₄	c ₃₅	c ₁₆	c ₂₆	c ₄₅

Table 8-14 (concluded) – Specification of mapping of idx to c_{ij} for 8x8 zig-zag and 8x8 field scan

idx	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
zig-zag	c ₇₂	c ₇₃	c ₆₄	c ₅₅	c ₄₆	c ₃₇	c ₄₇	c ₅₆	c ₆₅	c ₇₄	c ₇₅	c ₆₆	c ₅₇	c ₆₇	c ₇₆	c ₇₇
field	c ₅₅	c ₆₅	c ₇₅	c ₃₆	c ₀₇	c ₁₇	c ₄₆	c ₅₆	c ₆₆	c ₇₆	c ₂₇	c ₃₇	c ₄₇	c ₅₇	c ₆₇	c ₇₇

8.5.7 Derivation process for the chroma quantisation parameters and scaling function

Outputs of this process are:

- QP_C: the chroma quantisation parameter for each chroma component Cb and Cr
- QS_C: the additional chroma quantisation parameter for each chroma component Cb and Cr required for decoding SP and SI slices (if applicable)

NOTE 1 – QP quantisation parameter values QP_Y and QS_Y are always in the range of –QpBdOffset_Y to 51, inclusive. QP quantisation parameter values QP_C and QS_C are always in the range of –QpBdOffset_C to 51, inclusive.

The value of QP_C for a chroma component is determined from the current value of QP_Y and the value of chroma_qp_index_offset (for Cb) or second_chroma_qp_index_offset (for Cr).

NOTE 2 – The scaling equations are specified such that the equivalent transform coefficient level scaling factor doubles for every increment of 6 in QP_Y. Thus, there is an increase in the factor used for scaling of approximately 12 % for each increase of 1 in the value of QP_Y.

The value of QP_C for each chroma component is determined as specified in Table 8-15 based on the index denoted as qP_I.

The variable qP_{Offset} for each chroma component is derived as follows.

- If the chroma component is the Cb component, qP_{Offset} is specified as

$$qP_{\text{Offset}} = \text{chroma_qp_index_offset} \quad (8-307)$$

- Otherwise (the chroma component is the Cr component), qP_{Offset} is specified as

$$qP_{\text{Offset}} = \text{second_chroma_qp_index_offset} \quad (8-308)$$

The value of qP_I for each chroma component is derived as

$$qP_I = \text{Clip3}(-QpBdOffset_C, 51, QP_Y + qP_{\text{Offset}}) \quad (8-309)$$

The value of QP'_C for the chroma components is derived as

$$QP'_C = QP_C + QpBdOffset_C \quad (8-310)$$

The value of $BitDepth'_C$ for the chroma components is derived as

$$BitDepth'_C = BitDepth_C + residual_colour_transform_flag \quad (8-311)$$

Table 8-15 – Specification of QP_C as a function of qP_1

qP_1	<30	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
QP_C	$=qP_1$	29	30	31	32	32	33	34	34	35	35	36	36	37	37	37	38	38	38	39	39	39	39

When the current slice is an SP or SI slice, QS_C is derived using the above process, substituting QP_Y with QS_Y and QP_C with QS_C .

The function $LevelScale(m, i, j)$ is specified as follows.

- The 4x4 matrix $weightScale(i, j)$ is specified as follows.
 - The variable $mbIsInterFlag$ is derived as follows.
 - If the current macroblock is coded using Inter macroblock prediction modes, $mbIsInterFlag$ is set equal to 1.
 - Otherwise (the current macroblock is coded using Intra macroblock prediction modes), $mbIsInterFlag$ is set equal to 0.
 - The variable $iYCbCr$ derived as follows.
 - If the input array c relates to a luma residual block, $iYCbCr$ is set equal to 0.
 - Otherwise, if the input array c relates to a chroma residual block and the chroma component is equal to Cb, $iYCbCr$ is set equal to 1.
 - Otherwise (the input array c relates to a chroma residual block and the chroma component is equal to Cr), $iYCbCr$ is set equal to 2.
- The inverse scanning process for transform coefficients as specified in subclause 8.5.5 is invoked with $ScalingList4x4[iYCbCr + ((mbIsInterFlag == 1) ? 3 : 0)]$ as the input and the output is assigned to the 4x4 matrix $weightScale$.

$$LevelScale(m, i, j) = weightScale(i, j) * normAdjust(m, i, j) \quad (8-312)$$

where

$$normAdjust(m, i, j) = \begin{cases} v_{m0} & \text{for } (i \% 2, j \% 2) \text{ equal to } (0,0), \\ v_{m1} & \text{for } (i \% 2, j \% 2) \text{ equal to } (1,1), \\ v_{m2} & \text{otherwise;} \end{cases} \quad (8-313)$$

where the first and second subscripts of v are row and column indices, respectively, of the matrix specified as:

$$v = \begin{bmatrix} 10 & 16 & 13 \\ 11 & 18 & 14 \\ 13 & 20 & 16 \\ 14 & 23 & 18 \\ 16 & 25 & 20 \\ 18 & 29 & 23 \end{bmatrix}. \quad (8-314)$$

The function $\text{LevelScale8x8}(m, i, j)$ is specified as follows:

- The 8x8 matrix $\text{weightScale8x8}(i, j)$ is specified as follows.
 - The variable mbIsInterFlag is derived as follows.
 - If the current macroblock is coded using Inter macroblock prediction modes, mbIsInterFlag is set equal to 1.
 - Otherwise (the current macroblock is coded using Intra macroblock prediction modes), mbIsInterFlag is set equal to 0.
- The inverse scanning process for 8x8 luma transform coefficients as specified in subclause 8.5.6 is invoked with $\text{ScalingList8x8}[\text{mbIsInterFlag}]$ as the input and the output is assigned to the 8x8 matrix weightScale8x8 .

$$\text{LevelScale8x8}(m, i, j) = \text{weightScale8x8}(i, j) * \text{normAdjust8x8}(m, i, j) \quad (8-315)$$

where

$$\text{normAdjust8x8}(m, i, j) = \begin{cases} v_{m0} & \text{for } (i \% 4, j \% 4) \text{ equal to } (0,0), \\ v_{m1} & \text{for } (i \% 2, j \% 2) \text{ equal to } (1,1), \\ v_{m2} & \text{for } (i \% 4, j \% 4) \text{ equal to } (2,2), \\ v_{m3} & \text{for } (i \% 4, j \% 2) \text{ equal to } (0,1) \text{ or } (i \% 2, j \% 4) \text{ equal to } (1,0), \\ v_{m4} & \text{for } (i \% 4, j \% 4) \text{ equal to } (0,2) \text{ or } (i \% 4, j \% 4) \text{ equal to } (2,0), \\ v_{m5} & \text{otherwise;} \end{cases} \quad (8-316)$$

where the first and second subscripts of v are row and column indices, respectively, of the matrix specified as:

$$v = \begin{bmatrix} 20 & 18 & 32 & 19 & 25 & 24 \\ 22 & 19 & 35 & 21 & 28 & 26 \\ 26 & 23 & 42 & 24 & 33 & 31 \\ 28 & 25 & 45 & 26 & 35 & 33 \\ 32 & 28 & 51 & 30 & 40 & 38 \\ 36 & 32 & 58 & 34 & 46 & 43 \end{bmatrix}. \quad (8-317)$$

8.5.8 Scaling and transformation process for luma DC transform coefficients for Intra_16x16 macroblock type

Inputs to this process are transform coefficient level values for luma DC transform coefficients of Intra_16x16 macroblocks as a 4x4 array c with elements c_{ij} , where i and j form a two-dimensional frequency index.

Outputs of this process are 16 scaled DC values for luma 4x4 blocks of Intra_16x16 macroblocks as a 4x4 array dcY with elements dcY_{ij} .

Depending on the values of $\text{qpprime_y_zero_transform_bypass_flag}$ and QP'_Y , the following applies.

- If $\text{qpprime_y_zero_transform_bypass_flag}$ is equal to 1 and QP'_Y is equal to 0, the output dcY is derived as

$$\text{dcY}_{ij} = c_{ij} \text{ with } i, j = 0..3 \quad (8-318)$$

- Otherwise ($\text{qpprime_y_zero_transform_bypass_flag}$ is equal to 0 or QP'_Y is not equal to 0), the following text of this process specifies the output.

The inverse transform for the 4x4 luma DC transform coefficients is specified by:

$$f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}. \quad (8-319)$$

The bitstream shall not contain data that results in any element f_{ij} of f with $i, j = 0..3$ that exceeds the range of integer values from $-2^{(7 + \text{BitDepth}_Y)}$ to $2^{(7 + \text{BitDepth}_Y)} - 1$, inclusive.

After the inverse transform, scaling is performed as follows.

- If QP'_Y is greater than or equal to 36, the scaled result is derived as

$$dcY_{ij} = (f_{ij} * \text{LevelScale}(QP'_Y \% 6, 0, 0)) << (QP'_Y / 6 - 6), \quad \text{with } i, j = 0..3 \quad (8-320)$$

- Otherwise (QP'_Y is less than 36), the scaled result is derived as

$$dcY_{ij} = (f_{ij} * \text{LevelScale}(QP'_Y \% 6, 0, 0) + 2^{5 - QP'_Y / 6}) >> (6 - QP'_Y / 6), \quad \text{with } i, j = 0..3 \quad (8-321)$$

The bitstream shall not contain data that results in any element dcY_{ij} of dcY with $i, j = 0..3$ that exceeds the range of integer values from $-2^{(7 + \text{BitDepth}_Y)}$ to $2^{(7 + \text{BitDepth}_Y)} - 1$, inclusive.

NOTE 1 – When `entropy_coding_mode_flag` is equal to 0 and QP'_Y is less than 10 and `profile_idc` is equal to 66, 77, or 88, the range of values that can be represented for the elements c_{ij} of c is not sufficient to represent the full range of values of the elements dcY_{ij} of dcY that could be necessary to form a close approximation of the content of any possible source picture by use of the `Intra_16x16` macroblock type.

NOTE 2 – Since the range limit imposed on the elements dcY_{ij} of dcY is imposed after the right shift in Equation 8-321, a larger range of values must be supported in the decoder prior to the right shift.

8.5.9 Scaling and transformation process for chroma DC transform coefficients

Inputs to this process are transform coefficient level values for chroma DC transform coefficients of one chroma component of the macroblock as an $(\text{MbWidthC} / 4) \times (\text{MbHeightC} / 4)$ array c with elements c_{ij} , where i and j form a two-dimensional frequency index.

Outputs of this process are the scaled DC values as an $(\text{MbWidthC} / 4) \times (\text{MbHeightC} / 4)$ array dcC with elements dcC_{ij} .

Depending on the values of `qpprime_y_zero_transform_bypass_flag` and QP'_Y , the following applies.

- If `qpprime_y_zero_transform_bypass_flag` is equal to 1 and QP'_Y is equal to 0, the output dcC is derived as

$$dcC_{ij} = c_{ij} \quad \text{with } i = 0..(\text{MbWidthC} / 4) - 1 \text{ and } j = 0..(\text{MbHeightC} / 4) - 1. \quad (8-322)$$

- Otherwise (`qpprime_y_zero_transform_bypass_flag` is equal to 0 or QP'_Y is not equal to 0), the following text of this process specifies the output.

Depending on the variable `chroma_format_idc`, the inverse transform is specified as follows.

- If `chroma_format_idc` is equal to 1, the inverse transform for the 2x2 chroma DC transform coefficients is specified as

$$f = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (8-323)$$

- Otherwise, if `chroma_format_idc` is equal to 2, the inverse transform for the 2x4 chroma DC transform coefficients is specified as

$$f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \\ c_{20} & c_{21} \\ c_{30} & c_{31} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (8-324)$$

- Otherwise (chroma_format_idc is equal to 3), the inverse transform for the 4x4 chroma DC transform coefficients is specified as follows.
 - If residual_colour_transform_flag is equal to 1 and the current macroblock prediction mode MbPartPredMode(mb_type, 0) is Intra_4x4 or Intra_8x8, the inverse transform for the 4x4 chroma DC transform coefficients is specified as

$$f_{ij} = c_{ij} \ll 2 \text{ with } i, j = 0..3 \quad (8-325)$$

- Otherwise, the inverse transform for the 4x4 chroma DC transform coefficients is specified as

$$f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \quad (8-326)$$

The bitstream shall not contain data that results in any element f_{ij} of f with $i, j = 0..3$ that exceeds the range of integer values from $-2^{(7 + \text{BitDepth}_C)}$ to $2^{(7 + \text{BitDepth}_C)} - 1$, inclusive.

After the inverse transform, scaling is performed depending on the variable chroma_format_idc as follows.

- If chroma_format_idc is equal to 1, the scaled result is derived as

$$dcC_{ij} = ((f_{ij} * \text{LevelScale}(QP'_C \% 6, 0, 0)) \ll (QP'_C / 6)) \gg 5, \text{ with } i, j = 0, 1 \quad (8-327)$$

- If chroma_format_idc is equal to 2, the following applies.

- The variable $QP'_{C,DC}$ is derived as

$$QP'_{C,DC} = QP'_C + 3 \quad (8-328)$$

- Depending on the value of $QP'_{C,DC}$, the following applies.

- If $QP'_{C,DC}$ is greater than or equal to 36, the scaled result is derived as

$$dcC_{ij} = (f_{ij} * \text{LevelScale}(QP'_{C,DC} \% 6, 0, 0)) \ll (QP'_{C,DC} / 6 - 6), \text{ with } i = 0..3, j = 0, 1 \quad (8-329)$$

- Otherwise ($QP'_{C,DC}$ is less than 36), the scaled result is derived as

$$dcC_{ij} = (f_{ij} * \text{LevelScale}(QP'_{C,DC} \% 6, 0, 0) + 2^{5-QP'_{C,DC}/6}) \gg (6 - QP'_{C,DC} / 6), \text{ with } i = 0..3, j = 0, 1 \quad (8-330)$$

- Otherwise (chroma_format_idc is equal to 3), the following applies.

- If QP'_C is greater than or equal to 36, the scaled result is derived as

$$dcC_{ij} = (f_{ij} * \text{LevelScale}(QP'_C \% 6, 0, 0)) \ll (QP'_C / 6 - 6), \text{ with } i, j = 0..3. \quad (8-331)$$

- Otherwise (QP'_C is less than 36), the scaled result is derived as

$$dcC_{ij} = (f_{ij} * \text{LevelScale}(QP'_C \% 6, 0, 0) + 2^{5-QP'_C/6}) \gg (6 - QP'_C / 6), \text{ with } i, j = 0..3 \quad (8-332)$$

The bitstream shall not contain data that results in any element dcC_{ij} of dcC with $i, j = 0..3$ that exceeds the range of integer values from $-2^{(7 + \text{BitDepth}_C)}$ to $2^{(7 + \text{BitDepth}_C)} - 1$, inclusive.

NOTE 1 – When entropy_coding_mode_flag is equal to 0 and QP'_C is less than 4 and profile_idc is equal to 66, 77, or 88, the range of values that can be represented for the elements c_{ij} of c may not be sufficient to represent the full range of values of the elements dcC_{ij} of dcC that could be necessary to form a close approximation of the content of any possible source picture.

NOTE 2 – Since the range limit imposed on the elements dcC_{ij} of dcC is imposed after the right shift in Equation 8-327, 8-330, or 8-332, a larger range of values must be supported in the decoder prior to the right shift.

8.5.10 Scaling and transformation process for residual 4x4 blocks

Input to this process is a 4x4 array c with elements c_{ij} which is either an array relating to a residual block of the luma component or an array relating to a residual block of a chroma component.

Outputs of this process are residual sample values as 4x4 array r with elements r_{ij} .

Depending on the values of $qpprime_y_zero_transform_bypass_flag$ and QP'_Y , the following applies.

- If $qpprime_y_zero_transform_bypass_flag$ is equal to 1 and QP'_Y is equal to 0, the output r is derived as

$$r_{ij} = c_{ij} \text{ with } i, j = 0..3 \quad (8-333)$$

- Otherwise ($qpprime_y_zero_transform_bypass_flag$ is equal to 0 or QP'_Y is not equal to 0), the following text of this process specifies the output.

The variable $bitDepth$ is derived as follows.

- If the input array c relates to a luma residual block, $bitDepth$ is set equal to $BitDepth_Y$.
- Otherwise (the input array c relates to a chroma residual block), $bitDepth$ is set equal to $BitDepth'_C$.

The bitstream shall not contain data that results in any element c_{ij} of c with $i, j = 0..3$ that exceeds the range of integer values from $-2^{(7 + bitDepth)}$ to $2^{(7 + bitDepth)} - 1$, inclusive.

The variable $sMbFlag$ is derived as follows.

- If mb_type is equal to SI or the macroblock prediction mode is equal to Inter in an SP slice, $sMbFlag$ is set equal to 1,
- Otherwise (mb_type not equal to SI and the macroblock prediction mode is not equal to Inter in an SP slice), $sMbFlag$ is set equal to 0.

The variable qP is derived as follows.

- If the input array c relates to a luma residual block and $sMbFlag$ is equal to 0

$$qP = QP'_Y \quad (8-334)$$

- Otherwise, if the input array c relates to a luma residual block and $sMbFlag$ is equal to 1

$$qP = QS_Y \quad (8-335)$$

- Otherwise, if the input array c relates to a chroma residual block and $sMbFlag$ is equal to 0

$$qP = QP'_C \quad (8-336)$$

- Otherwise (the input array c relates to a chroma residual block and $sMbFlag$ is equal to 1),

$$qP = QS_C \quad (8-337)$$

Scaling of 4x4 block transform coefficient levels c_{ij} proceeds as follows.

- If all of the following conditions are true
 - i is equal to 0
 - j is equal to 0
 - c relates to a luma residual block coded using Intra_16x16 prediction mode or c relates to a chroma residual block

the variable d_{00} is derived by

$$d_{00} = c_{00} \quad (8-338)$$

– Otherwise, the following applies.

– If qP is greater than or equal to 24, the scaled result is derived as follows

$$d_{ij} = (c_{ij} * \text{LevelScale}(qP \% 6, i, j)) \ll (qP / 6 - 4), \text{ with } i, j = 0..3 \text{ except as noted above} \quad (8-339)$$

– Otherwise (qP is less than 24), the scaled result is derived as follows

$$d_{ij} = (c_{ij} * \text{LevelScale}(qP \% 6, i, j) + 2^{3-qP/6}) \gg (4 - qP / 6), \text{ with } i, j = 0..3 \text{ except as noted above} \quad (8-340)$$

The bitstream shall not contain data that results in any element d_{ij} of d with $i, j = 0..3$ that exceeds the range of integer values from $-2^{(7 + \text{bitDepth})}$ to $2^{(7 + \text{bitDepth})} - 1$, inclusive.

The transform process shall convert the block of scaled transform coefficients to a block of output samples in a manner mathematically equivalent to the following.

First, each (horizontal) row of scaled transform coefficients is transformed using a one-dimensional inverse transform as follows.

A set of intermediate values is computed as follows.

$$e_{i0} = d_{i0} + d_{i2}, \text{ with } i = 0..3 \quad (8-341)$$

$$e_{i1} = d_{i0} - d_{i2}, \text{ with } i = 0..3 \quad (8-342)$$

$$e_{i2} = (d_{i1} \gg 1) - d_{i3}, \text{ with } i = 0..3 \quad (8-343)$$

$$e_{i3} = d_{i1} + (d_{i3} \gg 1), \text{ with } i = 0..3 \quad (8-344)$$

The bitstream shall not contain data that results in any element e_{ij} of e with $i, j = 0..3$ that exceeds the range of integer values from $-2^{(7 + \text{bitDepth})}$ to $2^{(7 + \text{bitDepth})} - 1$, inclusive.

Then, the transformed result is computed from these intermediate values as follows.

$$f_{i0} = e_{i0} + e_{i3}, \text{ with } i = 0..3 \quad (8-345)$$

$$f_{i1} = e_{i1} + e_{i2}, \text{ with } i = 0..3 \quad (8-346)$$

$$f_{i2} = e_{i1} - e_{i2}, \text{ with } i = 0..3 \quad (8-347)$$

$$f_{i3} = e_{i0} - e_{i3}, \text{ with } i = 0..3 \quad (8-348)$$

The bitstream shall not contain data that results in any element f_{ij} of f with $i, j = 0..3$ that exceeds the range of integer values from $-2^{(7 + \text{bitDepth})}$ to $2^{(7 + \text{bitDepth})} - 1$, inclusive.

Then, each (vertical) column of the resulting matrix is transformed using the same one-dimensional inverse transform as follows.

A set of intermediate values is computed as follows.

$$g_{0j} = f_{0j} + f_{2j}, \text{ with } j = 0..3 \quad (8-349)$$

$$g_{1j} = f_{0j} - f_{2j}, \text{ with } j = 0..3 \quad (8-350)$$

$$g_{2j} = (f_{1j} \gg 1) - f_{3j}, \text{ with } j = 0..3 \quad (8-351)$$

$$g_{3j} = f_{1j} + (f_{3j} \gg 1), \text{ with } j = 0..3 \quad (8-352)$$

The bitstream shall not contain data that results in any element g_{ij} of g with $i, j = 0..3$ that exceeds the range of integer values from $-2^{(7 + \text{bitDepth})}$ to $2^{(7 + \text{bitDepth})} - 1$, inclusive.

Then, the transformed result is computed from these intermediate values as follows.

$$h_{0j} = g_{0j} + g_{3j}, \text{ with } j = 0..3 \quad (8-353)$$

$$h_{1j} = g_{1j} + g_{2j}, \text{ with } j = 0..3 \quad (8-354)$$

$$h_{2j} = g_{1j} - g_{2j}, \text{ with } j = 0..3 \quad (8-355)$$

$$h_{3j} = g_{0j} - g_{3j}, \text{ with } j = 0..3 \quad (8-356)$$

The bitstream shall not contain data that results in any element h_{ij} of h with $i, j = 0..3$ that exceeds the range of integer values from $-2^{(7 + \text{bitDepth})}$ to $2^{(7 + \text{bitDepth})} - 33$, inclusive.

After performing both the one-dimensional horizontal and the one-dimensional vertical inverse transforms to produce an array of transformed samples, the final constructed residual sample values is derived as

$$r_{ij} = (h_{ij} + 2^5) \gg 6 \text{ with } i, j = 0..3 \quad (8-357)$$

8.5.11 Scaling and transformation process for residual 8x8 luma blocks

Input to this process is an 8x8 array c with elements c_{ij} which is an array relating to an 8x8 residual block of the luma component.

Outputs of this process are residual sample values as 8x8 array r with elements r_{ij} .

Depending on the values of `qpprime_y_zero_transform_bypass_flag` and QP'_Y , the following applies.

- If `qpprime_y_zero_transform_bypass_flag` is equal to 1 and QP'_Y is equal to 0, the output r is derived as

$$r_{ij} = c_{ij} \text{ with } i, j = 0..7 \quad (8-358)$$

- Otherwise (`qpprime_y_zero_transform_bypass_flag` is equal to 0 or QP'_Y is not equal to 0), the following text of this process specifies the output.

The bitstream shall not contain data that results in any element c_{ij} of c with $i, j = 0..7$ that exceeds the range of integer values from $-2^{(7 + \text{BitDepth}_Y)}$ to $2^{(7 + \text{BitDepth}_Y)} - 1$, inclusive.

The scaling process for 8x8 block transform coefficient levels c_{ij} proceeds as follows.

- If QP'_Y is greater than or equal to 36, the scaled result is derived as

$$d_{ij} = (c_{ij} * \text{LevelScale8x8}(QP'_Y \% 6, i, j)) \ll (QP'_Y / 6 - 6), \text{ with } i, j = 0..7 \quad (8-359)$$

- Otherwise (QP'_Y is less than 36), the scaled result is derived as

$$d_{ij} = (c_{ij} * \text{LevelScale8x8}(QP'_Y \% 6, i, j)) + 2^{5-QP'_Y/6} \gg (6 - QP'_Y/6), \text{ with } i, j = 0..7 \quad (8-360)$$

The bitstream shall not contain data that results in any element d_{ij} of d with $i, j = 0..7$ that exceeds the range of integer values from $-2^{(7 + \text{BitDepth}_Y)}$ to $2^{(7 + \text{BitDepth}_Y)} - 1$, inclusive.

The transform process shall convert the block of scaled transform coefficients to a block of output samples in a manner mathematically equivalent to the following.

First, each (horizontal) row of scaled transform coefficients is transformed using a one-dimensional inverse transform as follows.

- A set of intermediate values e_{ij} is derived by

$$e_{i0} = d_{i0} + d_{i4}, \text{ with } i = 0..7 \quad (8-361)$$

$$e_{i1} = -d_{i3} + d_{i5} - d_{i7} - (d_{i7} \gg 1), \text{ with } i = 0..7 \quad (8-362)$$

$$e_{i2} = d_{i0} - d_{i4}, \text{ with } i = 0..7 \quad (8-363)$$

$$e_{i3} = d_{i1} + d_{i7} - d_{i3} - (d_{i3} \gg 1), \text{ with } i = 0..7 \quad (8-364)$$

$$e_{i4} = (d_{i2} \gg 1) - d_{i6}, \text{ with } i = 0..7 \quad (8-365)$$

$$e_{i5} = -d_{i1} + d_{i7} + d_{i5} + (d_{i5} \gg 1), \text{ with } i = 0..7 \quad (8-366)$$

$$e_{i6} = d_{i2} + (d_{i6} \gg 1), \text{ with } i = 0..7 \quad (8-367)$$

$$e_{i7} = d_{i3} + d_{i5} + d_{i1} + (d_{i1} \gg 1), \text{ with } i = 0..7 \quad (8-368)$$

- A second set of intermediate results f_{ij} is computed from the intermediate values e_{ij} as

$$f_{i0} = e_{i0} + e_{i6}, \text{ with } i = 0..7 \quad (8-369)$$

$$f_{i1} = e_{i1} + (e_{i7} \gg 2), \text{ with } i = 0..7 \quad (8-370)$$

$$f_{i2} = e_{i2} + e_{i4}, \text{ with } i = 0..7 \quad (8-371)$$

$$f_{i3} = e_{i3} + (e_{i5} \gg 2), \text{ with } i = 0..7 \quad (8-372)$$

$$f_{i4} = e_{i2} - e_{i4}, \text{ with } i = 0..7 \quad (8-373)$$

$$f_{i5} = (e_{i3} \gg 2) - e_{i5}, \text{ with } i = 0..7 \quad (8-374)$$

$$f_{i6} = e_{i0} - e_{i6}, \text{ with } i = 0..7 \quad (8-375)$$

$$f_{i7} = e_{i7} - (e_{i1} \gg 2), \text{ with } i = 0..7 \quad (8-376)$$

- Then, the transformed result g_{ij} is computed from these intermediate values f_{ij} as

$$g_{i0} = f_{i0} + f_{i7}, \text{ with } i = 0..7 \quad (8-377)$$

$$g_{i1} = f_{i2} + f_{i5}, \text{ with } i = 0..7 \quad (8-378)$$

$$g_{i2} = f_{i4} + f_{i3}, \text{ with } i = 0..7 \quad (8-379)$$

$$g_{i3} = f_{i6} + f_{i1}, \text{ with } i = 0..7 \quad (8-380)$$

$$g_{i4} = f_{i6} - f_{i1}, \text{ with } i = 0..7 \quad (8-381)$$

$$g_{i5} = f_{i4} - f_{i3}, \text{ with } i = 0..7 \quad (8-382)$$

$$g_{i6} = f_{i2} - f_{i5}, \text{ with } i = 0..7 \quad (8-383)$$

$$g_{i7} = f_{i0} - f_{i7}, \text{ with } i = 0..7 \quad (8-384)$$

Then, each (vertical) column of the resulting matrix is transformed using the same one-dimensional inverse transform as follows.

- A set of intermediate values h_{ij} is computed from the horizontally transformed value g_{ij} as

$$h_{0j} = g_{0j} + g_{4j}, \text{ with } j = 0..7 \quad (8-385)$$

$$h_{1j} = -g_{3j} + g_{5j} - g_{7j} - (g_{7j} >> 1), \text{ with } j = 0..7 \quad (8-386)$$

$$h_{2j} = g_{0j} - g_{4j}, \text{ with } j = 0..7 \quad (8-387)$$

$$h_{3j} = g_{1j} + g_{7j} - g_{3j} - (g_{3j} >> 1), \text{ with } j = 0..7 \quad (8-388)$$

$$h_{4j} = (g_{2j} >> 1) - g_{6j}, \text{ with } j = 0..7 \quad (8-389)$$

$$h_{5j} = -g_{1j} + g_{7j} + g_{5j} + (g_{5j} >> 1), \text{ with } j = 0..7 \quad (8-390)$$

$$h_{6j} = g_{2j} + (g_{6j} >> 1), \text{ with } j = 0..7 \quad (8-391)$$

$$h_{7j} = g_{3j} + g_{5j} + g_{1j} + (g_{1j} >> 1), \text{ with } j = 0..7 \quad (8-392)$$

- A second set of intermediate results k_{ij} is computed from the intermediate values h_{ij} as

$$k_{0j} = h_{0j} + h_{6j}, \text{ with } j = 0..7 \quad (8-393)$$

$$k_{1j} = h_{1j} + (h_{7j} >> 2), \text{ with } j = 0..7 \quad (8-394)$$

$$k_{2j} = h_{2j} + h_{4j}, \text{ with } j = 0..7 \quad (8-395)$$

$$k_{3j} = h_{3j} + (h_{5j} >> 2), \text{ with } j = 0..7 \quad (8-396)$$

$$k_{4j} = h_{2j} - h_{4j}, \text{ with } j = 0..7 \quad (8-397)$$

$$k_{5j} = (h_{3j} >> 2) - h_{5j}, \text{ with } j = 0..7 \quad (8-398)$$

$$k_{6j} = h_{0j} - h_{6j}, \text{ with } j = 0..7 \quad (8-399)$$

$$k_{7j} = h_{7j} - (h_{1j} \gg 2), \text{ with } j = 0..7 \quad (8-400)$$

- Then, the transformed result m_{ij} is computed from these intermediate values k_{ij} as

$$m_{0j} = k_{0j} + k_{7j}, \text{ with } j = 0..7 \quad (8-401)$$

$$m_{1j} = k_{2j} + k_{5j}, \text{ with } j = 0..7 \quad (8-402)$$

$$m_{2j} = k_{4j} + k_{3j}, \text{ with } j = 0..7 \quad (8-403)$$

$$m_{3j} = k_{6j} + k_{1j}, \text{ with } j = 0..7 \quad (8-404)$$

$$m_{4j} = k_{6j} - k_{1j}, \text{ with } j = 0..7 \quad (8-405)$$

$$m_{5j} = k_{4j} - k_{3j}, \text{ with } j = 0..7 \quad (8-406)$$

$$m_{6j} = k_{2j} - k_{5j}, \text{ with } j = 0..7 \quad (8-407)$$

$$m_{7j} = k_{0j} - k_{7j}, \text{ with } j = 0..7 \quad (8-408)$$

The bitstream shall not contain data that results in any element e_{ij} , f_{ij} , g_{ij} , h_{ij} , or k_{ij} for i and j in the range of $0..7$, inclusive, that exceeds the range of integer values from $-2^{(7 + \text{BitDepth}_Y)}$ to $2^{(7 + \text{BitDepth}_Y)} - 1$, inclusive.

The bitstream shall not contain data that results in any element m_{ij} for i and j in the range of $0..7$, inclusive, that exceeds the range of integer values from $-2^{(7 + \text{BitDepth}_Y)}$ to $2^{(7 + \text{BitDepth}_Y)} - 33$, inclusive.

After performing both the one-dimensional horizontal and the one-dimensional vertical inverse transforms to produce an array of transformed samples, the final constructed residual sample values are derived as

$$r_{ij} = (m_{ij} + 2^5) \gg 6 \text{ with } i, j = 0..7 \quad (8-409)$$

8.5.12 Picture construction process prior to deblocking filter process

Inputs to this process are

- luma4x4BlkIdx or chroma4x4BlkIdx or luma8x8BlkIdx
- a sample array u with elements u_{ij} which is either a 4x4 luma block or a 4x4 chroma block or an 8x8 luma block

The position of the upper-left luma sample of the current macroblock is derived by invoking the inverse macroblock scanning process in subclause 6.4.1 with CurrMbAddr as input and the output being assigned to (xP, yP) .

When u is a luma block, for each sample u_{ij} of the luma block, the following applies.

- Depending on the size of the block u , the following applies.
 - If u is an 4x4 luma block, the position of the upper-left sample of the 4x4 luma block with index luma4x4BlkIdx inside the macroblock is derived by invoking the inverse 4x4 luma block scanning process in subclause 6.4.3 with luma4x4BlkIdx as the input and the output being assigned to (xO, yO) , and the variable nE is set equal to 4.
 - Otherwise (u is an 8x8 luma block), the position of the upper-left sample of the 8x8 luma block with index luma8x8BlkIdx inside the macroblock is derived by invoking the inverse 8x8 luma block scanning process in subclause 6.4.4 with luma8x8BlkIdx as the input and the output being assigned to (xO, yO) , and the variable nE is set equal to 8.
- Depending on the variable MbaffFrameFlag and the current macroblock, the following applies.
 - If MbaffFrameFlag is equal to 1 and the current macroblock is a field macroblock

$$S'_L[xP + xO + j, yP + 2 * (yO + i)] = u_{ij} \text{ with } i, j = 0..nE - 1 \quad (8-410)$$

- Otherwise (MbaffFrameFlag is equal to 0 or the current macroblock is a frame macroblock),

$$S'_L[xP + xO + j, yP + yO + i] = u_{ij} \quad \text{with } i, j = 0..nE - 1 \quad (8-411)$$

When u is a chroma block, for each sample u_{ij} of the 4x4 chroma block, the following applies.

- The subscript C in the variable S'_C is replaced with Cb for the Cb chroma component and with Cr for the Cr chroma component.
- Depending on the variable chroma_format_idc, the position of the upper-left sample of a 4x4 chroma block with index chroma4x4BlkIdx inside the macroblock is derived as follows.

- If chroma_format_idc is equal to 1 or 2, the following applies.

$$xO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 0) \quad (8-412)$$

$$yO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 1) \quad (8-413)$$

- Otherwise (chroma_format_idc is equal to 3), the following applies.

$$xO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx} / 4, 8, 8, 16, 0) + \text{InverseRasterScan}(\text{chroma4x4BlkIdx} \% 4, 4, 4, 8, 0) \quad (8-414)$$

$$yO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx} / 4, 8, 8, 16, 1) + \text{InverseRasterScan}(\text{chroma4x4BlkIdx} \% 4, 4, 4, 8, 1) \quad (8-415)$$

- Depending on the variable MbaffFrameFlag and the current macroblock, the following applies.

- If MbaffFrameFlag is equal to 1 and the current macroblock is a field macroblock

$$S'_C[(xP / \text{subWidthC}) + xO + j, ((yP + \text{SubHeightC} - 1) / \text{SubHeightC}) + 2 * (yO + i)] = u_{ij} \quad \text{with } i, j = 0..3 \quad (8-416)$$

- Otherwise (MbaffFrameFlag is equal to 0 or the current macroblock is a frame macroblock),

$$S'_C[(xP / \text{subWidthC}) + xO + j, (yP / \text{SubHeightC}) + yO + i] = u_{ij} \quad \text{with } i, j = 0..3 \quad (8-417)$$

8.5.13 Residual colour transform process

This process is invoked when residual_colour_transform_flag is equal to 1.

After invoking, this process is suspended until the derivation of $R_{Y,ij}$, $R_{Cb,ij}$, and $R_{Cr,ij}$ has been completed for $i, j = 0..ijMax$, where $ijMax$ is specified as follows.

- If transform_size_8x8_flag is equal to 0, the variable $ijMax$ is set equal to 3.
- Otherwise (transform_size_8x8_flag is equal to 1), the variable $ijMax$ is set equal to 7.

At the resumption of this process, all values $R_{Y,ij}$, $R_{Cb,ij}$, and $R_{Cr,ij}$ with $i, j = 0..ijMax$ shall be available through prior invocations of the relevant processes specified in subclauses 8.5.1, 8.5.2, 8.5.3, or 8.5.4

For each $i, j = 0..ijMax$, the residual colour transform is computed as

$$t = R_{Y,ij} - (R_{Cb,ij} >> 1) \quad (8-418)$$

$$R_{G,ij} = t + R_{Cb,ij} \quad (8-419)$$

$$R_{B,ij} = t - (R_{Cr,ij} >> 1) \quad (8-420)$$

$$R_{R,ij} = R_{B,ij} + R_{Cr,ij} \quad (8-421)$$

NOTE – The residual colour transform is similar to the YCgCo transformation specified in Equations E-30 through E-33. However, the residual colour transform operates on the decoded residual difference data within the decoding process rather than operating as a post-processing step that is outside the decoding process specified in this Recommendation | International Standard.

8.6 Decoding process for P macroblocks in SP slices or SI macroblocks

This process is invoked when decoding P macroblock types in an SP slice type or an SI macroblock type in SI slices.

Inputs to this process are the prediction residual transform coefficient levels and the predicted samples for the current macroblock.

Outputs of this process are the decoded samples of the current macroblock prior to the deblocking filter process.

This subclause specifies the transform coefficient decoding process and picture construction process for P macroblock types in SP slices and SI macroblock type in SI slices.

NOTE – SP slices make use of Inter predictive coding to exploit temporal redundancy in the sequence, in a similar manner to P slice coding. Unlike P slice coding, however, SP slice coding allows identical reconstruction of a slice even when different reference pictures are being used. SI slices make use of spatial prediction, in a similar manner to I slices. SI slice coding allows identical reconstruction to a corresponding SP slice. The properties of SP and SI slices aid in providing functionalities for bitstream switching, splicing, random access, fast-forward, fast reverse, and error resilience/recovery.

An SP slice consists of macroblocks coded either as I macroblock types or P macroblock types.

An SI slice consists of macroblocks coded either as I macroblock types or SI macroblock type.

The transform coefficient decoding process and picture construction process prior to deblocking filter process for I macroblock types in SI slices is invoked as specified in subclause 8.5. SI macroblock type is decoded as described below.

When the current macroblock is coded as P_Skip, all values of LumaLevel, ChromaDCLevel, ChromaACLevel are set equal to 0 for the current macroblock.

8.6.1 SP decoding process for non-switching pictures

This process is invoked, when decoding P macroblock types in SP slices in which `sp_for_switch_flag` is equal to 0.

Inputs to this process are Inter prediction samples for the current macroblock from subclause 8.4 and the prediction residual transform coefficient levels.

Outputs of this process are the decoded samples of the current macroblock prior to the deblocking filter process.

This subclause applies to all macroblocks in SP slices in which `sp_for_switch_flag` is equal to 0, except those with macroblock prediction mode equal to `Intra_4x4` or `Intra_16x16`. It does not apply to SI slices.

8.6.1.1 Luma transform coefficient decoding process

Inputs to this process are Inter prediction luma samples for the current macroblock pred_L from subclause 8.4 and the prediction residual transform coefficient levels, LumaLevel, and the index of the 4x4 luma block `luma4x4BlkIdx`.

The position of the upper-left sample of the 4x4 luma block with index `luma4x4BlkIdx` inside the current macroblock is derived by invoking the inverse 4x4 luma block scanning process in subclause 6.4.3 with `luma4x4BlkIdx` as the input and the output being assigned to (x, y) .

Let the variable p be a 4x4 array of prediction samples with element p_{ij} being derived as follows.

$$p_{ij} = \text{pred}_L[x + j, y + i] \quad \text{with } i, j = 0..3 \quad (8-422)$$

The variable p is transformed producing transform coefficients c^p according to:

$$c^p = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ p_{30} & p_{31} & p_{32} & p_{33} \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \quad (8-423)$$

The inverse transform coefficient scanning process as described in subclause 8.5.5 is invoked with $\text{LumaLevel}[\text{ luma4x4BlkIdx }]$ as the input and the two-dimensional array c^r with elements c_{ij}^r as the output.

The prediction residual transform coefficients c^r are scaled using quantisation parameter QP_Y , and added to the transform coefficients of the prediction block c^p with $i, j = 0..3$ as follows.

$$c_{ij}^s = c_{ij}^p + (((c_{ij}^r * \text{LevelScale}(QP_Y \% 6, i, j) * A_{ij}) << (QP_Y / 6)) >> 10) \quad (8-424)$$

where $\text{LevelScale}(m, i, j)$ is specified in Equation 8-312, and where A_{ij} is specified as:

$$A_{ij} = \begin{cases} 16 & \text{for } (i, j) \in \{(0,0), (0,2), (2,0), (2,2)\}, \\ 25 & \text{for } (i, j) \in \{(1,1), (1,3), (3,1), (3,3)\}, \\ 20 & \text{otherwise;} \end{cases} \quad (8-425)$$

The function $\text{LevelScale2}(m, i, j)$, used in the formulas below, is specified as:

$$\text{LevelScale2}(m, i, j) = \begin{cases} w_{m0} & \text{for } (i, j) \in \{(0,0), (0,2), (2,0), (2,2)\}, \\ w_{m1} & \text{for } (i, j) \in \{(1,1), (1,3), (3,1), (3,3)\}, \\ w_{m2} & \text{otherwise;} \end{cases} \quad (8-426)$$

where the first and second subscripts of w are row and column indices, respectively, of the matrix specified as:

$$w = \begin{bmatrix} 13107 & 5243 & 8066 \\ 11916 & 4660 & 7490 \\ 10082 & 4194 & 6554 \\ 9362 & 3647 & 5825 \\ 8192 & 3355 & 5243 \\ 7282 & 2893 & 4559 \end{bmatrix} \quad (8-427)$$

The resulting sum, c^s , is quantised with a quantisation parameter QS_Y and with $i, j = 0..3$ as follows.

$$c_{ij} = \text{Sign}(c_{ij}^s) * ((\text{Abs}(c_{ij}^s) * \text{LevelScale2}(QS_Y \% 6, i, j) + (1 << (14 + QS_Y / 6))) >> (15 + QS_Y / 6)) \quad (8-428)$$

The scaling and transformation process for residual 4x4 blocks as specified in subclause 8.5.10 is invoked with c as the input and r as the output.

The 4x4 array u with elements u_{ij} is derived as follows.

$$u_{ij} = \text{Clip1}_Y(r_{ij}) \text{ with } i, j = 0..3 \quad (8-429)$$

The picture construction process prior to deblocking filter process in subclause 8.5.12 is invoked with luma4x4BlkIdx and u as the inputs.

8.6.1.2 Chroma transform coefficient decoding process

Inputs to this process are Inter prediction chroma samples for the current macroblock from subclause 8.4 and the prediction residual transform coefficient levels, ChromaDCLevel and ChromaACLevel .

This process is invoked twice: once for the Cb component and once for the Cr component. The component is referred to by replacing C with Cb for the Cb component and C with Cr for the Cr component. Let $iCbCr$ select the current chroma component.

For each 4x4 block of the current chroma component indexed using chroma4x4BlkIdx with chroma4x4BlkIdx equal to 0..3, the following applies.

- The position of the upper-left sample of a 4x4 chroma block with index chroma4x4BlkIdx inside the macroblock is derived as follows.

$$x = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 0) \quad (8-430)$$

$$y = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 1) \quad (8-431)$$

- Let p be a 4x4 array of prediction samples with elements p_{ij} being derived as follows.

$$p_{ij} = \text{pred}_c[x + j, y + i] \quad \text{with } i, j = 0..3 \quad (8-432)$$

- The 4x4 array p is transformed producing transform coefficients $c^p(\text{chroma4x4BlkIdx})$ using Equation 8-423.
- The variable chromaList , which is a list of 16 entries, is derived. $\text{chromaList}[0]$ is set equal to 0. $\text{chromaList}[k]$ with index $k = 1..15$ are specified as follows.

$$\text{chromaList}[k] = \text{ChromaACLevel}[\text{iCbCr}][\text{chroma4x4BlkIdx}][k - 1] \quad (8-433)$$

- The inverse transform coefficient scanning process as described in subclause 8.5.5 is invoked with chromaList as the input and the 4x4 array c^r as the output.
- The prediction residual transform coefficients c^r are scaled using quantisation parameter QP_C , and added to the transform coefficients of the prediction block c^p with $i, j = 0..3$ except for the combination $i = 0, j = 0$ as follows.

$$c_{ij}^s = c_{ij}^p(\text{chroma4x4BlkIdx}) + ((c_{ij}^r * \text{LevelScale}(QP_C \% 6, i, j) * A_{ij}) << (QP_C / 6)) >> 10) \quad (8-434)$$

- The resulting sum, c^s , is quantised with a quantisation parameter QS_C and with $i, j = 0..3$ except for the combination $i = 0, j = 0$ as follows. The derivation of $c_{00}(\text{chroma4x4BlkIdx})$ is described below in this subclause.

$$c_{ij}(\text{chroma4x4BlkIdx}) = (\text{Sign}(c_{ij}^s) * (\text{Abs}(c_{ij}^s) * \text{LevelScale2}(QS_C \% 6, i, j) + (1 << (14 + QS_C / 6)))) >> (15 + QS_C / 6) \quad (8-435)$$

- The scaling and transformation process for residual 4x4 blocks as specified in 8.5.10 is invoked with $c(\text{chroma4x4BlkIdx})$ as the input and r as the output.
- The 4x4 array u with elements u_{ij} is derived as follows.

$$u_{ij} = \text{Clip1}_c(r_{ij}) \quad \text{with } i, j = 0..3 \quad (8-436)$$

- The picture construction process prior to deblocking filter process in subclause 8.5.12 is invoked with chroma4x4BlkIdx and u as the inputs.

The derivation of the DC transform coefficient level $c_{00}(\text{chroma4x4BlkIdx})$ is specified as follows. The DC transform coefficients of the 4 prediction chroma 4x4 blocks of the current component of the macroblock are assembled into a 2x2 matrix with elements $c_{00}^p(\text{chroma4x4BlkIdx})$ and a 2x2 transform is applied to the DC transform coefficients as follows

$$dc^p = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00}^p(0) & c_{00}^p(1) \\ c_{00}^p(2) & c_{00}^p(3) \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (8-437)$$

The chroma DC prediction residual transform coefficient levels, $\text{ChromaDCLevel}[\text{iCbCr}][k]$ with $k = 0..3$ are scaled using quantisation parameter QP , and added to the prediction DC transform coefficients as follows.

$$dc_{ij}^s = dc_{ij}^p + ((\text{ChromaDCLevel}[\text{iCbCr}][j * 2 + i] * \text{LevelScale}(QP_C \% 6, 0, 0) * A_{00}) << (QP_C / 6)) >> 9) \quad \text{with } i, j = 0, 1 \quad (8-438)$$

The 2x2 array dc^s , is quantised using the quantisation parameter QS_C as follows.

$$dc_{ij}^r = (\text{Sign}(dc_{ij}^s) * (\text{Abs}(dc_{ij}^s) * \text{LevelScale2}(QS_C \% 6, 0, 0) + (1 << (15 + QS_C / 6)))) >> (16 + QS_C / 6) \quad \text{with } i, j = 0, 1 \quad (8-439)$$

The 2x2 array f with elements f_{ij} and $i, j = 0..1$ is derived as follows.

$$f = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} dc_{00}^r & dc_{01}^r \\ dc_{10}^r & dc_{11}^r \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \quad (8-440)$$

Scaling of the elements f_{ij} of f is performed as follows.

$$c_{00}(j * 2 + i) = ((f_{ij} * \text{LevelScale}(QS_C \% 6, 0, 0)) \ll (QS_C / 6)) \gg 5 \text{ with } i, j = 0, 1 \quad (8-441)$$

8.6.2 SP and SI slice decoding process for switching pictures

This process is invoked, when decoding P macroblock types in SP slices in which `sp_for_switch_flag` is equal to 1 and when decoding SI macroblock type in SI slices.

Inputs to this process are the prediction residual transform coefficient levels and the prediction sample arrays pred_L , pred_{Cb} and pred_{Cr} for the current macroblock.

8.6.2.1 Luma transform coefficient decoding process

Inputs to this process are prediction luma samples pred_L and the luma prediction residual transform coefficient levels, `LumaLevel`.

The 4x4 array p with elements p_{ij} with $i, j = 0..3$ is derived as in subclause 8.6.1.1, is transformed according to Equation 8-423 to produce transform coefficients c^p . These transform coefficients are then quantised with the quantisation parameter QS_Y , as follows:

$$c_{ij}^s = \text{Sign}(c_{ij}^p) * ((\text{Abs}(c_{ij}^p) * \text{LevelScale2}(QS_Y \% 6, i, j) + (1 \ll (14 + QS_Y / 6))) \gg (15 + QS_Y / 6)) \text{ with } i, j = 0..3 \quad (8-442)$$

The inverse transform coefficient scanning process as described in subclause 8.5.5 is invoked with `LumaLevel[luma4x4BlkIdx]` as the input and the two-dimensional array c^r with elements c_{ij}^r as the output.

The 4x4 array c with elements c_{ij} with $i, j = 0..3$ is derived as follows.

$$c_{ij} = c_{ij}^r + c_{ij}^s \text{ with } i, j = 0..3 \quad (8-443)$$

The scaling and transformation process for residual 4x4 blocks as specified in subclause 8.5.10 is invoked with c as the input and r as the output.

The 4x4 array u with elements u_{ij} is derived as follows.

$$u_{ij} = \text{Clip1}_Y(r_{ij}) \text{ with } i, j = 0..3 \quad (8-444)$$

The picture construction process prior to deblocking filter process in subclause 8.5.12 is invoked with `luma4x4BlkIdx` and u as the inputs.

8.6.2.2 Chroma transform coefficient decoding process

Inputs to this process are predicted chroma samples for the current macroblock from subclause 8.4 and the prediction residual transform coefficient levels, `ChromaDCLevel` and `ChromaACLevel`.

This process is invoked twice: once for the Cb component and once for the Cr component. The component is referred to by replacing C with Cb for the Cb component and C with Cr for the Cr component. Let `iCbCr` select the current chroma component.

For each 4x4 block of the current chroma component indexed using `chroma4x4BlkIdx` with `chroma4x4BlkIdx` equal to 0..3, the following applies.

1. The 4x4 array p with elements p_{ij} with $i, j = 0..3$ is derived as in subclause 8.6.1.2, is transformed according to Equation 8-423 to produce transform coefficients $c^p(\text{chroma4x4BlkIdx})$. These transform coefficients are then quantised with the quantisation parameter QS_C , with $i, j = 0..3$ except for the combination $i = 0, j = 0$ as follows. The processing of $c_{00}^p(\text{chroma4x4BlkIdx})$ is described below in this subclause.

$$c_{ij}^s = (\text{Sign}(c_{ij}^p(\text{chroma4x4BlkIdx})) * (\text{Abs}(c_{ij}^p(\text{chroma4x4BlkIdx})) * \text{LevelScale2}(\text{QSC} \% 6, i, j) + (1 << (14 + \text{QSC} / 6)))) >> (15 + \text{QSC} / 6) \quad (8-445)$$

- The variable chromaList, which is a list of 16 entries, is derived. chromaList[0] is set equal to 0. chromaList[k] with index k = 1..15 are specified as follows.

$$\text{chromaList}[k] = \text{ChromaACLevel}[\text{iCbCr}][\text{chroma4x4BlkIdx}][k - 1] \quad (8-446)$$

- The inverse transform coefficient scanning process as described in subclause 8.5.5 is invoked with chromaList as the input and the two-dimensional array $c^r(\text{chroma4x4BlkIdx})$ with elements $c_{ij}^r(\text{chroma4x4BlkIdx})$ as the output.
- The 4x4 array $c(\text{chroma4x4BlkIdx})$ with elements $c_{ij}(\text{chroma4x4BlkIdx})$ with $i, j = 0..3$ except for the combination $i = 0, j = 0$ is derived as follows. The derivation of $c_{00}(\text{chroma4x4BlkIdx})$ is described below.

$$c_{ij}(\text{chroma4x4BlkIdx}) = c_{ij}^r(\text{chroma4x4BlkIdx}) + c_{ij}^s \quad (8-447)$$

- The scaling and transformation process for residual 4x4 blocks as specified in subclause 8.5.10 is invoked with $c(\text{chroma4x4BlkIdx})$ as the input and r as the output.
- The 4x4 array u with elements u_{ij} is derived as follows.

$$u_{ij} = \text{Clip1}_C(r_{ij}) \text{ with } i, j = 0..3 \quad (8-448)$$

- The picture construction process prior to deblocking filter process in subclause 8.5.12 is invoked with chroma4x4BlkIdx and u as the inputs.

The derivation of the DC transform coefficient level $c_{00}(\text{chroma4x4BlkIdx})$ is specified as follows. The DC transform coefficients of the 4 prediction 4x4 chroma blocks of the current component of the macroblock, $c_{00}^p(\text{chroma4x4BlkIdx})$, are assembled into a 2x2 matrix, and a 2x2 transform is applied to the DC transform coefficients of these blocks according to Equation 8-437 resulting in DC transform coefficients dc_{ij}^p .

These DC transform coefficients are then quantised with the quantisation parameter QSC , as given by:

$$dc_{ij}^s = (\text{Sign}(dc_{ij}^p) * (\text{Abs}(dc_{ij}^p) * \text{LevelScale2}(\text{QSC} \% 6, 0, 0) + (1 << (15 + \text{QSC} / 6)))) >> (16 + \text{QSC} / 6) \quad \text{with } i, j = 0, 1 \quad (8-449)$$

The parsed chroma DC prediction residual transform coefficients, $\text{ChromaDCLevel}[\text{iCbCr}][k]$ with $k = 0..3$ are added to these quantised DC transform coefficients of the prediction block, as given by:

$$dc_{ij}^r = dc_{ij}^s + \text{ChromaDCLevel}[\text{iCbCr}][j * 2 + i] \text{ with } i, j = 0, 1 \quad (8-450)$$

The 2x2 array f with elements f_{ij} and $i, j = 0..1$ is derived using Equation 8-440.

The 2x2 array f with elements f_{ij} and $i, j = 0..1$ is copied as follows.

$$c_{00}(j * 2 + i) = f_{ij} \text{ with } i, j = 0, 1 \quad (8-451)$$

8.7 Deblocking filter process

A conditional filtering process is applied to all NxN (where $N = 4$ or $N = 8$ for luma, and $N = 4$ for chroma) block edges of a picture, except edges at the boundary of the picture and any edges for which the deblocking filter process is disabled by `disable_deblocking_filter_idc`, as specified below. This filtering process is performed on a macroblock basis after the completion of the picture construction process prior to deblocking filter process (as specified in subclauses 8.5 and 8.6) for the entire decoded picture, with all macroblocks in a picture processed in order of increasing macroblock addresses.

NOTE 1 – Prior to the operation of the deblocking filter process for each macroblock, the deblocked samples of the macroblock or macroblock pair above (if any) and the macroblock or macroblock pair to the left (if any) of the current macroblock are always available because the deblocking filter process is performed after the completion of the picture construction process prior to deblocking filter process for the entire decoded picture. However, for purposes of determining which edges are to be filtered

when `disable_deblocking_filter_idc` is equal to 2, macroblocks in different slices are considered not available during specified steps of the operation of the deblocking filter process.

The deblocking filter process is invoked for the luma and chroma components separately. For each macroblock and each component, vertical edges are filtered first, starting with the edge on the left-hand side of the macroblock proceeding through the edges towards the right-hand side of the macroblock in their geometrical order, and then horizontal edges are filtered, starting with the edge on the top of the macroblock proceeding through the edges towards the bottom of the macroblock in their geometrical order. Figure 8-10 shows edges of a macroblock which can be interpreted as luma or chroma edges.

When interpreting the edges in Figure 8-10 as luma edges, depending on the `transform_size_8x8_flag`, the following applies.

- If `transform_size_8x8_flag` is equal to 0, both types, the solid bold and dashed bold luma edges are filtered.
- Otherwise (`transform_size_8x8_flag` is equal to 1), only the solid bold luma edges are filtered.

When interpreting the edges in Figure 8-10 as chroma edges, depending on `chroma_format_idc`, the following applies.

- If `chroma_format_idc` is equal to 1 (4:2:0 format), only the solid bold chroma edges are filtered.
- Otherwise, if `chroma_format_idc` is equal to 2 (4:2:2 format), the solid bold vertical chroma edges are filtered and both types, the solid bold and dashed bold horizontal chroma edges are filtered.
- Otherwise, if `chroma_format_idc` is equal to 3 (4:4:4 format), both types, the solid bold and dashed bold chroma edges are filtered.
- Otherwise (`chroma_format_idc` is equal to 0 (monochrome)), no chroma edges are filtered.

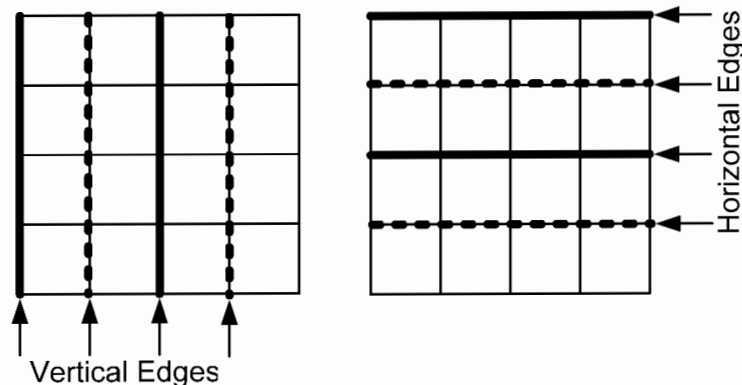


Figure 8-10 – Boundaries in a macroblock to be filtered

For the current macroblock address `CurrMbAddr` proceeding over values $0..PicSizeInMbs - 1$, the following applies.

1. The derivation process for neighbouring macroblocks specified in subclause 6.4.8.1 is invoked and the output is assigned to `mbAddrA` and `mbAddrB`.
2. The variables `fieldModeMbFlag`, `filterInternalEdgesFlag`, `filterLeftMbEdgeFlag` and `filterTopMbEdgeFlag` are derived as follows.
 - The variable `fieldModeMbFlag` is derived as follows.
 - If any of the following conditions is true, `fieldModeMbFlag` is set equal to 1.
 - `field_pic_flag` is equal to 1
 - `MbaffFrameFlag` is equal to 1 and the macroblock `CurrMbAddr` is a field macroblock
 - Otherwise, `fieldModeMbFlag` is set equal to 0.
 - The variable `filterInternalEdgesFlag` is derived as follows.

- If `disable_deblocking_filter_idc` for the slice that contains the macroblock `CurrMbAddr` is equal to 1, the variable `filterInternalEdgesFlag` is set equal to 0;
 - Otherwise (`disable_deblocking_filter_idc` for the slice that contains the macroblock `CurrMbAddr` is not equal to 1), the variable `filterInternalEdgesFlag` is set equal to 1.
 - The variable `filterLeftMbEdgeFlag` is derived as follows.
 - If any of the following conditions is true, the variable `filterLeftMbEdgeFlag` is set equal to 0.
 - `MbaffFrameFlag` is equal to 0 and `CurrMbAddr % PicWidthInMbs` is equal to 0.
 - `MbaffFrameFlag` is equal to 1 and $(CurrMbAddr \gg 1) \% PicWidthInMbs$ is equal to 0
 - `disable_deblocking_filter_idc` for the slice that contains the macroblock `CurrMbAddr` is equal to 1
 - `disable_deblocking_filter_idc` for the slice that contains the macroblock `CurrMbAddr` is equal to 2 and the macroblock `mbAddrA` is not available
 - Otherwise, the variable `filterLeftMbEdgeFlag` is set equal to 1.
 - The variable `filterTopMbEdgeFlag` is derived as follows.
 - If any of the following conditions is true, the variable `filterTopMbEdgeFlag` is set equal to 0.
 - `MbaffFrameFlag` is equal to 0 and `CurrMbAddr` is less than `PicWidthInMbs`.
 - `MbaffFrameFlag` is equal to 1, $(CurrMbAddr \gg 1)$ is less than `PicWidthInMbs`, and the macroblock `CurrMbAddr` is a field macroblock.
 - `MbaffFrameFlag` is equal to 1, $(CurrMbAddr \gg 1)$ is less than `PicWidthInMbs`, the macroblock `CurrMbAddr` is a frame macroblock, and `CurrMbAddr % 2` is equal to 0.
 - `disable_deblocking_filter_idc` for the slice that contains the macroblock `CurrMbAddr` is equal to 1
 - `disable_deblocking_filter_idc` for the slice that contains the macroblock `CurrMbAddr` is equal to 2 and the macroblock `mbAddrB` is not available
 - Otherwise, the variable `filterTopMbEdgeFlag` is set equal to 1.
3. Given the variables `fieldModeMbFlag`, `filterInternalEdgesFlag`, `filterLeftMbEdgeFlag` and `filterTopMbEdgeFlag` the deblocking filtering is controlled as follows.
- When `filterLeftMbEdgeFlag` is equal to 1, the filtering of the left vertical luma edge is specified as follows.
 - The process specified in subclause 8.7.1 is invoked with `chromaEdgeFlag` = 0, `verticalEdgeFlag` = 1, `fieldModeFilteringFlag` = `fieldModeMbFlag`, and $(xE_k, yE_k) = (0, k)$ with $k = 0..15$ as input and S'_L as output.
 - When `filterInternalEdgesFlag` is equal to 1, the filtering of the internal vertical luma edges is specified as follows.
 - When `transform_size_8x8_flag` is equal to 0, the process specified in subclause 8.7.1 is invoked with `chromaEdgeFlag` = 0, `verticalEdgeFlag` = 1, `fieldModeFilteringFlag` = `fieldModeMbFlag`, and $(xE_k, yE_k) = (4, k)$ with $k = 0..15$ as input and S'_L as output.
 - The process specified in subclause 8.7.1 is invoked with `chromaEdgeFlag` = 0, `verticalEdgeFlag` = 1, `fieldModeFilteringFlag` = `fieldModeMbFlag`, and $(xE_k, yE_k) = (8, k)$ with $k = 0..15$ as input and S'_L as output.
 - When `transform_size_8x8_flag` is equal to 0, the process specified in subclause 8.7.1 is invoked with `chromaEdgeFlag` = 0, `verticalEdgeFlag` = 1, `fieldModeFilteringFlag` = `fieldModeMbFlag`, and $(xE_k, yE_k) = (12, k)$ with $k = 0..15$ as input and S'_L as output.
 - When `filterTopMbEdgeFlag` is equal to 1, the filtering of the top horizontal luma edge is specified as follows.
 - If `MbaffFrameFlag` is equal to 1, $(CurrMbAddr \% 2)$ is equal to 0, `CurrMbAddr` is greater than or equal to $2 * PicWidthInMbs$, the macroblock `CurrMbAddr` is a frame macroblock, and the macroblock $(CurrMbAddr - 2 * PicWidthInMbs + 1)$ is a field macroblock, the following applies.
 - The process specified in subclause 8.7.1 is invoked with `chromaEdgeFlag` = 0, `verticalEdgeFlag` = 0, `fieldModeFilteringFlag` = 1, and $(xE_k, yE_k) = (k, 0)$ with $k = 0..15$ as input and S'_L as output.

- The process specified in subclause 8.7.1 is invoked with $\text{chromaEdgeFlag} = 0$, $\text{verticalEdgeFlag} = 0$, $\text{fieldModeFilteringFlag} = 1$, and $(xE_k, yE_k) = (k, 1)$ with $k = 0..15$ as input and S'_L as output.
 - Otherwise, the process specified in subclause 8.7.1 is invoked with $\text{chromaEdgeFlag} = 0$, $\text{verticalEdgeFlag} = 0$, $\text{fieldModeFilteringFlag} = \text{fieldModeMbFlag}$, and $(xE_k, yE_k) = (k, 0)$ with $k = 0..15$ as input and S'_L as output.
- When $\text{filterInternalEdgesFlag}$ is equal to 1, the filtering of the internal horizontal luma edges is specified as follows.
 - When $\text{transform_size_8x8_flag}$ is equal to 0, the process specified in subclause 8.7.1 is invoked with $\text{chromaEdgeFlag} = 0$, $\text{verticalEdgeFlag} = 0$, $\text{fieldModeFilteringFlag} = \text{fieldModeMbFlag}$, and $(xE_k, yE_k) = (k, 4)$ with $k = 0..15$ as input and S'_L as output.
 - The process specified in subclause 8.7.1 is invoked with $\text{chromaEdgeFlag} = 0$, $\text{verticalEdgeFlag} = 0$, $\text{fieldModeFilteringFlag} = \text{fieldModeMbFlag}$, and $(xE_k, yE_k) = (k, 8)$ with $k = 0..15$ as input and S'_L as output.
 - When $\text{transform_size_8x8_flag}$ is equal to 0, the process specified in subclause 8.7.1 is invoked with $\text{chromaEdgeFlag} = 0$, $\text{verticalEdgeFlag} = 0$, $\text{fieldModeFilteringFlag} = \text{fieldModeMbFlag}$, and $(xE_k, yE_k) = (k, 12)$ with $k = 0..15$ as input and S'_L as output.
- For the filtering of both chroma components with $iCbCr = 0$ for Cb and $iCbCr = 1$ for Cr, the following applies.
 - When $\text{filterLeftMbEdgeFlag}$ is equal to 1, the filtering of the left vertical chroma edge is specified as follows.
 - The process specified in subclause 8.7.1 is invoked with $\text{chromaEdgeFlag} = 1$, $iCbCr$, $\text{verticalEdgeFlag} = 1$, $\text{fieldModeFilteringFlag} = \text{fieldModeMbFlag}$, and $(xE_k, yE_k) = (0, k)$ with $k = 0..\text{MbHeightC} - 1$ as input and S'_C with C being replaced by Cb for $iCbCr = 0$ and C being replaced by Cr for $iCbCr = 1$ as output.
 - When $\text{filterInternalEdgesFlag}$ is equal to 1, the filtering of the internal vertical chroma edge is specified as follows.
 - The process specified in subclause 8.7.1 is invoked with $\text{chromaEdgeFlag} = 1$, $iCbCr$, $\text{verticalEdgeFlag} = 1$, $\text{fieldModeFilteringFlag} = \text{fieldModeMbFlag}$, and $(xE_k, yE_k) = (4, k)$ with $k = 0..\text{MbHeightC} - 1$ as input and S'_C with C being replaced by Cb for $iCbCr = 0$ and C being replaced by Cr for $iCbCr = 1$ as output.
 - When chroma_format_idc is equal to 3, the process specified in subclause 8.7.1 is invoked with $\text{chromaEdgeFlag} = 1$, $iCbCr$, $\text{verticalEdgeFlag} = 1$, $\text{fieldModeFilteringFlag} = \text{fieldModeMbFlag}$, and $(xE_k, yE_k) = (8, k)$ with $k = 0..\text{MbHeightC} - 1$ as input and S'_C with C being replaced by Cb for $iCbCr = 0$ and C being replaced by Cr for $iCbCr = 1$ as output.
 - When chroma_format_idc is equal to 3, the process specified in subclause 8.7.1 is invoked with $\text{chromaEdgeFlag} = 1$, $iCbCr$, $\text{verticalEdgeFlag} = 1$, $\text{fieldModeFilteringFlag} = \text{fieldModeMbFlag}$, and $(xE_k, yE_k) = (12, k)$ with $k = 0..\text{MbHeightC} - 1$ as input and S'_C with C being replaced by Cb for $iCbCr = 0$ and C being replaced by Cr for $iCbCr = 1$ as output.
- When $\text{filterTopMbEdgeFlag}$ is equal to 1, the filtering of the top horizontal chroma edge is specified as follows.
 - If MbaffFrameFlag is equal to 1, $(\text{CurrMbAddr} \% 2)$ is equal to 0, CurrMbAddr is greater than or equal to $2 * \text{PicWidthInMbs}$, the macroblock CurrMbAddr is a frame macroblock, and the macroblock $(\text{CurrMbAddr} - 2 * \text{PicWidthInMbs} + 1)$ is a field macroblock, the following applies.
 - The process specified in subclause 8.7.1 is invoked with $\text{chromaEdgeFlag} = 1$, $iCbCr$, $\text{verticalEdgeFlag} = 0$, $\text{fieldModeFilteringFlag} = 1$, and $(xE_k, yE_k) = (k, 0)$ with $k = 0..\text{MbWidthC} - 1$ as input and S'_C with C being replaced by Cb for $iCbCr = 0$ and C being replaced by Cr for $iCbCr = 1$ as output.
 - The process specified in subclause 8.7.1 is invoked with $\text{chromaEdgeFlag} = 1$, $iCbCr$, $\text{verticalEdgeFlag} = 0$, $\text{fieldModeFilteringFlag} = 1$, and $(xE_k, yE_k) = (k, 1)$ with $k = 0..\text{MbWidthC} - 1$ as input and S'_C with C being replaced by Cb for $iCbCr = 0$ and C being replaced by Cr for $iCbCr = 1$ as output.

- Otherwise, the process specified in subclause 8.7.1 is invoked with $\text{chromaEdgeFlag} = 1$, iCbCr , $\text{verticalEdgeFlag} = 0$, $\text{fieldModeFilteringFlag} = \text{fieldModeMbFlag}$, and $(x_{E_k}, y_{E_k}) = (k, 0)$ with $k = 0..MbWidthC - 1$ as input and S'_C with C being replaced by C_b for $\text{iCbCr} = 0$ and C being replaced by C_r for $\text{iCbCr} = 1$ as output.
- When $\text{filterInternalEdgesFlag}$ is equal to 1, the filtering of the internal horizontal chroma edge is specified as follows.
 - The process specified in subclause 8.7.1 is invoked with $\text{chromaEdgeFlag} = 1$, iCbCr , $\text{verticalEdgeFlag} = 0$, $\text{fieldModeFilteringFlag} = \text{fieldModeMbFlag}$, and $(x_{E_k}, y_{E_k}) = (k, 4)$ with $k = 0..MbWidthC - 1$ as input and S'_C with C being replaced by C_b for $\text{iCbCr} = 0$ and C being replaced by C_r for $\text{iCbCr} = 1$ as output.
 - When chroma_format_idc is not equal to 1, the process specified in subclause 8.7.1 is invoked with $\text{chromaEdgeFlag} = 1$, iCbCr , $\text{verticalEdgeFlag} = 0$, $\text{fieldModeFilteringFlag} = \text{fieldModeMbFlag}$, and $(x_{E_k}, y_{E_k}) = (k, 8)$ with $k = 0..MbWidthC - 1$ as input and S'_C with C being replaced by C_b for $\text{iCbCr} = 0$ and C being replaced by C_r for $\text{iCbCr} = 1$ as output.
 - When chroma_format_idc is not equal to 1, the process specified in subclause 8.7.1 is invoked with $\text{chromaEdgeFlag} = 1$, iCbCr , $\text{verticalEdgeFlag} = 0$, $\text{fieldModeFilteringFlag} = \text{fieldModeMbFlag}$, and $(x_{E_k}, y_{E_k}) = (k, 12)$ with $k = 0..MbWidthC - 1$ as input and S'_C with C being replaced by C_b for $\text{iCbCr} = 0$ and C being replaced by C_r for $\text{iCbCr} = 1$ as output.

NOTE 2 – When field mode filtering ($\text{fieldModeFilteringFlag}$ is equal to 1) is applied across the top horizontal edges of a frame macroblock, this vertical filtering across the top or bottom macroblock boundary may involve some samples that extend across an internal block edge that is also filtered internally in frame mode.

NOTE 3 – For example, in 4:2:0 chroma format when $\text{transform_size_8x8_flag}$ is equal to 0, the following applies. 3 horizontal luma edges, 1 horizontal chroma edge for C_b , and 1 horizontal chroma edge for C_r are filtered that are internal to a macroblock. When field mode filtering ($\text{fieldModeFilteringFlag}$ is equal to 1) is applied to the top edges of a frame macroblock, 2 horizontal luma, 2 horizontal chroma edges for C_b , and 2 horizontal chroma edges for C_r between the frame macroblock and the above macroblock pair are filtered using field mode filtering, for a total of up to 5 horizontal luma edges, 3 horizontal chroma edges for C_b , and 3 horizontal chroma edges for C_r filtered that are considered to be controlled by the frame macroblock. In all other cases, at most 4 horizontal luma, 2 horizontal chroma edges for C_b , and 2 horizontal chroma edges for C_r are filtered that are considered to be controlled by a particular macroblock.

Finally, the arrays S'_L , S'_{Cb} , S'_{Cr} are assigned to the arrays S_L , S_{Cb} , S_{Cr} (which represent the decoded picture), respectively.

8.7.1 Filtering process for block edges

Inputs to this process are chromaEdgeFlag , the chroma component index iCbCr (when chromaEdgeFlag is equal to 1), verticalEdgeFlag , $\text{fieldModeFilteringFlag}$, and a set of nE sample locations (x_{E_k}, y_{E_k}) , with $k = 0..nE - 1$, expressed relative to the upper left corner of the macroblock CurrMbAddr . The set of sample locations (x_{E_k}, y_{E_k}) represent the sample locations immediately to the right of a vertical edge (when verticalEdgeFlag is equal to 1) or immediately below a horizontal edge (when verticalEdgeFlag is equal to 0).

The variable nE is derived as follows.

- If chromaEdgeFlag is equal to 0, nE is set equal to 16.
- Otherwise (chromaEdgeFlag is equal to 1), nE is set equal to $(\text{verticalEdgeFlag} == 1) ? MbHeightC : MbWidthC$.

Let s' be a variable specifying a luma or chroma sample array, be derived as follows.

- If chromaEdgeFlag is equal to 0, s' represents the luma sample array S'_L of the current picture.
- Otherwise, if chromaEdgeFlag is equal to 1 and iCbCr is equal to 0, s' represents the chroma sample array S'_{Cb} of the chroma component C_b of the current picture.
- Otherwise (chromaEdgeFlag is equal to 1 and iCbCr is equal to 1), s' represents the chroma sample array S'_{Cr} of the chroma component C_r of the current picture.

The variable dy is derived as follows.

- If $\text{fieldModeFilteringFlag}$ is equal to 1 and MbaffFrameFlag is equal to 1, dy is set equal to 2.
- Otherwise ($\text{fieldModeFilteringFlag}$ is equal to 0 or MbaffFrameFlag is equal to 0), dy is set equal to 1.

The position of the upper-left luma sample of the macroblock CurrMbAddr is derived by invoking the inverse macroblock scanning process in subclause 6.4.1 with mbAddr = CurrMbAddr as input and the output being assigned to (xI, yI).

The variables xP and yP are derived as follows.

- If chromaEdgeFlag is equal to 0, xP is set equal to xI and yP is set equal to yI.
- Otherwise (chromaEdgeFlag is equal to 1), xP is set equal to xI / SubWidthC and yP is set equal to (yI + SubHeightC – 1) / SubHeightC.

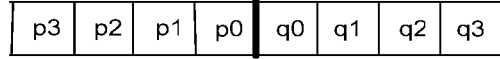


Figure 8-11 – Convention for describing samples across a 4x4 block horizontal or vertical boundary

For each sample location (xE_k, yE_k), k = 0 .. nE – 1, the following applies.

- The filtering process is applied to a set of eight samples across a 4x4 block horizontal or vertical edge denoted as p_i and q_i with i = 0..3 as shown in Figure 8-11 with the edge lying between p₀ and q₀. p_i and q_i with i = 0..3 are specified as follows.

- If verticalEdgeFlag is equal to 1,

$$q_i = s'[xP + xE_k + i, yP + dy * yE_k] \quad (8-452)$$

$$p_i = s'[xP + xE_k - i - 1, yP + dy * yE_k] \quad (8-453)$$

- Otherwise (verticalEdgeFlag is equal to 0),

$$q_i = s'[xP + xE_k, yP + dy * (yE_k + i) - (yE_k \% 2)] \quad (8-454)$$

$$p_i = s'[xP + xE_k, yP + dy * (yE_k - i - 1) - (yE_k \% 2)] \quad (8-455)$$

- The process specified in subclause 8.7.2 is invoked with the sample values p_i and q_i (i = 0..3), chromaEdgeFlag, verticalEdgeFlag, and fieldModeFilteringFlag as input, and the output is assigned to the filtered result sample values p'_i and q'_i with i = 0..2.
- The input sample values p_i and q_i with i = 0..2 are replaced by the corresponding filtered result sample values p'_i and q'_i with i = 0..2 inside the sample array s' as follows.

- If verticalEdgeFlag is equal to 1,

$$s'[xP + xE_k + i, yP + dy * yE_k] = q'_i \quad (8-456)$$

$$s'[xP + xE_k - i - 1, yP + dy * yE_k] = p'_i \quad (8-457)$$

- Otherwise (verticalEdgeFlag is equal to 0),

$$s'[xP + xE_k, yP + dy * (yE_k + i) - (yE_k \% 2)] = q'_i \quad (8-458)$$

$$s'[xP + xE_k, yP + dy * (yE_k - i - 1) - (yE_k \% 2)] = p'_i \quad (8-459)$$

8.7.2 Filtering process for a set of samples across a horizontal or vertical block edge

Inputs to this process are the input sample values p_i and q_i with i in the range of 0..3 of a single set of samples across an edge that is to be filtered, `chromaEdgeFlag`, `verticalEdgeFlag`, and `fieldModeFilteringFlag`.

Outputs of this process are the filtered result sample values p'_i and q'_i with i in the range of 0..2.

The content dependent boundary filtering strength variable bS is derived as follows.

- If `chromaEdgeFlag` is equal to 0, the derivation process for the content dependent boundary filtering strength specified in subclause 8.7.2.1 is invoked with p_0 , q_0 , and `verticalEdgeFlag` as input, and the output is assigned to bS .
- Otherwise (`chromaEdgeFlag` is equal to 1), the bS used for filtering a set of samples of a horizontal or vertical chroma edge is set equal to the value of bS for filtering the set of samples of a horizontal or vertical luma edge, respectively, that contains the luma sample at location (`SubWidthC * x`, `SubHeightC * y`) inside the luma array of the same field, where (x , y) is the location of the chroma sample q_0 inside the chroma array for that field.

The process specified in subclause 8.7.2.2 is invoked with p_0 , q_0 , p_1 , q_1 , `chromaEdgeFlag`, and bS as input, and the output is assigned to `filterSamplesFlag`, `indexA`, α , and β .

Depending on the variable `filterSamplesFlag`, the following applies.

- If `filterSamplesFlag` is equal to 1, the following applies.
 - If bS is less than 4, the process specified in subclause 8.7.2.3 is invoked with p_i and q_i ($i = 0..2$), `chromaEdgeFlag`, bS , β , and `indexA` given as input, and the output is assigned to p'_i and q'_i ($i = 0..2$).
 - Otherwise (bS is equal to 4), the process specified in subclause 8.7.2.4 is invoked with p_i and q_i ($i = 0..3$), `chromaEdgeFlag`, α , and β given as input, and the output is assigned to p'_i and q'_i ($i = 0..2$).
- Otherwise (`filterSamplesFlag` is equal to 0), the filtered result samples p'_i and q'_i ($i = 0..2$) are replaced by the corresponding input samples p_i and q_i :

$$\text{for } i = 0..2, \quad p'_i = p_i \quad (8-460)$$

$$\text{for } i = 0..2, \quad q'_i = q_i \quad (8-461)$$

8.7.2.1 Derivation process for the luma content dependent boundary filtering strength

Inputs to this process are the input sample values p_0 and q_0 of a single set of samples across an edge that is to be filtered and `verticalEdgeFlag`.

Output of this process is the variable bS .

Let the variable `mixedModeEdgeFlag` be derived as follows.

- If `MbaffFrameFlag` is equal to 1 and the samples p_0 and q_0 are in different macroblock pairs, one of which is a field macroblock pair and the other is a frame macroblock pair, `mixedModeEdgeFlag` is set equal to 1
- Otherwise, `mixedModeEdgeFlag` is set equal to 0.

The variable bS is derived as follows.

- If the block edge is also a macroblock edge and any of the following conditions are true, a value of bS equal to 4 is the output:
 - the samples p_0 and q_0 are both in frame macroblocks and either or both of the samples p_0 or q_0 is in a macroblock coded using an Intra macroblock prediction mode
 - the samples p_0 and q_0 are both in frame macroblocks and either or both of the samples p_0 or q_0 is in a macroblock that is in a slice with `slice_type` equal to SP or SI
 - `MbaffFrameFlag` is equal to 1 or `field_pic_flag` is equal to 1, and `verticalEdgeFlag` is equal to 1, and either or both of the samples p_0 or q_0 is in a macroblock coded using an Intra macroblock prediction mode
 - `MbaffFrameFlag` is equal to 1 or `field_pic_flag` is equal to 1, and `verticalEdgeFlag` is equal to 1, and either or both of the samples p_0 or q_0 is in a macroblock that is in a slice with `slice_type` equal to SP or SI

- Otherwise, if any of the following conditions are true, a value of bS equal to 3 is the output:
 - mixedModeEdgeFlag is equal to 0 and either or both of the samples p_0 or q_0 is in a macroblock coded using an Intra macroblock prediction mode
 - mixedModeEdgeFlag is equal to 0 and either or both of the samples p_0 or q_0 is in a macroblock that is in a slice with slice_type equal to SP or SI
 - mixedModeEdgeFlag is equal to 1, verticalEdgeFlag is equal to 0, and either or both of the samples p_0 or q_0 is in a macroblock coded using an Intra macroblock prediction mode
 - mixedModeEdgeFlag is equal to 1, verticalEdgeFlag is equal to 0, and either or both of the samples p_0 or q_0 is in a macroblock that is in a slice with slice_type equal to SP or SI
- Otherwise, if the following condition is true, a value of bS equal to 2 is the output:
 - the luma block containing sample p_0 or the luma block containing sample q_0 contains non-zero transform coefficient levels
- Otherwise, if any of the following conditions are true, a value of bS equal to 1 is the output:
 - mixedModeEdgeFlag is equal to 1
 - mixedModeEdgeFlag is equal to 0 and for the prediction of the macroblock/sub-macroblock partition containing the sample p_0 different reference pictures or a different number of motion vectors are used than for the prediction of the macroblock/sub-macroblock partition containing the sample q_0 .
 NOTE 1 – The determination of whether the reference pictures used for the two macroblock/sub-macroblock partitions are the same or different is based only on which pictures are referenced, without regard to whether a prediction is formed using an index into reference picture list 0 or an index into reference picture list 1, and also without regard to whether or not the index position within a reference picture list is different or not.
 - mixedModeEdgeFlag is equal to 0 and one motion vector is used to predict the macroblock/sub-macroblock partition containing the sample p_0 and one motion vector is used to predict the macroblock/sub-macroblock partition containing the sample q_0 and the absolute difference between the horizontal or vertical component of the motion vectors used is greater than or equal to 4 in units of quarter luma frame samples.
 - mixedModeEdgeFlag is equal to 0 and two motion vectors and two different reference pictures are used to predict the macroblock/sub-macroblock partition containing the sample p_0 and two motion vectors for the same two reference pictures are used to predict the macroblock/sub-macroblock partition containing the sample q_0 and the absolute difference between the horizontal or vertical component of the two motion vectors used in the prediction of the two macroblock/sub-macroblock partitions for the same reference picture is greater than or equal to 4 in units of quarter luma frame samples.
 - mixedModeEdgeFlag is equal to 0 and two motion vectors for the same reference picture are used to predict the macroblock/sub-macroblock partition containing the sample p_0 and two motion vectors for the same reference picture are used to predict the macroblock/sub-macroblock partition containing the sample q_0 and both of the following conditions are true:
 - The absolute difference between the horizontal or vertical component of list 0 motion vectors used in the prediction of the two macroblock/sub-macroblock partitions is greater than or equal to 4 in quarter luma frame samples or the absolute difference between the horizontal or vertical component of the list 1 motion vectors used in the prediction of the two macroblock/sub-macroblock partitions is greater than or equal to 4 in units of quarter luma frame samples.
 - The absolute difference between the horizontal or vertical component of list 0 motion vector used in the prediction of the macroblock/sub-macroblock partition containing the sample p_0 and the list 1 motion vector used in the prediction of the macroblock/sub-macroblock partition containing the sample q_0 is greater than or equal to 4 in units of quarter luma frame samples or the absolute difference between the horizontal or vertical component of the list 1 motion vector used in the prediction of the macroblock/sub-macroblock partition containing the sample p_0 and list 0 motion vector used in the prediction of the macroblock/sub-macroblock partition containing the sample q_0 is greater than or equal to 4 in units of quarter luma frame samples.

NOTE 2 – A vertical difference of 4 in units of quarter luma frame samples is a difference of 2 in units of quarter luma field samples

- Otherwise, a value of bS equal to 0 is the output.

8.7.2.2 Derivation process for the thresholds for each block edge

Inputs to this process are the input sample values p_0 , q_0 , p_1 and q_1 of a single set of samples across an edge that is to be filtered, chromaEdgeFlag, and bS, for the set of input samples, as specified in 8.7.2.

Outputs of this process are the variable filterSamplesFlag, which indicates whether the input samples are filtered, the value of indexA, and the values of the threshold variables α and β .

Let qP_p and qP_q be variables specifying quantisation parameter values for the macroblocks containing the samples p_0 and q_0 , respectively. The variables qP_z (with z being replaced by p or q) are derived as follows.

- If chromaEdgeFlag is equal to 0, the following applies.
 - If the macroblock containing the sample z_0 is an I_PCM macroblock, qP_z is set to 0.
 - Otherwise (the macroblock containing the sample z_0 is not an I_PCM macroblock), qP_z is set to the value of QP_Y of the macroblock containing the sample z_0 .
- Otherwise (chromaEdgeFlag is equal to 1), the following applies.
 - If the macroblock containing the sample z_0 is an I_PCM macroblock, qP_z is set to the value of QP_C that corresponds to a value of 0 for QP_Y as specified in subclause 8.5.7.
 - Otherwise (the macroblock containing the sample z_0 is not an I_PCM macroblock), qP_z is set to the value of QP_C that corresponds to the value QP_Y of the macroblock containing the sample z_0 as specified in subclause 8.5.7.

Let qP_{av} be a variable specifying an average quantisation parameter. It is derived as follows.

$$qP_{av} = (qP_p + qP_q + 1) \gg 1 \quad (8-462)$$

NOTE – In SP and SI slices, qP_{av} is derived in the same way as in other slice types. QS_Y from Equation 7-28 is not used in the deblocking filter.

Let indexA be a variable that is used to access the α table (Table 8-16) as well as the t_{C0} table (Table 8-17), which is used in filtering of edges with bS less than 4 as specified in subclause 8.7.2.3, and let indexB be a variable that is used to access the β table (Table 8-16). The variables indexA and indexB are derived as follows, where the values of FilterOffsetA and FilterOffsetB are the values of those variables specified in subclause 7.4.3 for the slice that contains the macroblock containing sample q_0 .

$$\text{indexA} = \text{Clip3}(0, 51, qP_{av} + \text{FilterOffsetA}) \quad (8-463)$$

$$\text{indexB} = \text{Clip3}(0, 51, qP_{av} + \text{FilterOffsetB}) \quad (8-464)$$

The variables α' and β' depending on the values of indexA and indexB are specified in Table 8-16. Depending on chromaEdgeFlag, the corresponding threshold variables α and β are derived as follows.

- If chromaEdgeFlag is equal to 0,

$$\alpha = \alpha' * (1 \ll (\text{BitDepth}_Y - 8)) \quad (8-465)$$

$$\beta = \beta' * (1 \ll (\text{BitDepth}_Y - 8)) \quad (8-466)$$

- Otherwise (chromaEdgeFlag is equal to 1),

$$\alpha = \alpha' * (1 \ll (\text{BitDepth}_C - 8)) \quad (8-467)$$

$$\beta = \beta' * (1 \ll (\text{BitDepth}_C - 8)) \quad (8-468)$$

The variable filterSamplesFlag is derived by

$$\text{filterSamplesFlag} = (bS \neq 0 \ \&\& \ \text{Abs}(p_0 - q_0) < \alpha \ \&\& \ \text{Abs}(p_1 - p_0) < \beta \ \&\& \ \text{Abs}(q_1 - q_0) < \beta) \quad (8-469)$$

Table 8-16 – Derivation of offset dependent threshold variables α' and β' from indexA and indexB

	indexA (for α') or indexB (for β')																									
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
α'	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	5	6	7	8	9	10	12	13
β'	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	3	3	3	3	4	4	4

Table 8-16 (concluded) – Derivation of indexA and indexB from offset dependent threshold variables α' and β'

	indexA (for α') or indexB (for β')																									
	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
α'	15	17	20	22	25	28	32	36	40	45	50	56	63	71	80	90	101	113	127	144	162	182	203	226	255	255
β'	6	6	7	7	8	8	9	9	10	10	11	11	12	12	13	13	14	14	15	15	16	16	17	17	18	18

8.7.2.3 Filtering process for edges with bS less than 4

Inputs to this process are the input sample values p_i and q_i ($i = 0..2$) of a single set of samples across an edge that is to be filtered, chromaEdgeFlag, bS, β , and indexA, for the set of input samples, as specified in 8.7.2.

Outputs of this process are the filtered result sample values p'_i and q'_i ($i = 0..2$) for the set of input sample values.

The filtered result samples p'_0 and q'_0 are derived by

$$\Delta = \text{Clip3}(-t_c, t_c, (((q_0 - p_0) < 2) + (p_1 - q_1) + 4) >> 3)) \quad (8-470)$$

$$p'_0 = \text{Clip1}(p_0 + \Delta) \quad (8-471)$$

$$q'_0 = \text{Clip1}(q_0 - \Delta) \quad (8-472)$$

where the threshold t_c is determined as follows.

- If chromaEdgeFlag is equal to 0,

$$t_c = t_{c0} + ((a_p < \beta) ? 1 : 0) + ((a_q < \beta) ? 1 : 0) \quad (8-473)$$

- Otherwise (chromaEdgeFlag is equal to 1),

$$t_c = t_{c0} + 1 \quad (8-474)$$

Depending on the values of indexA and bS the variable t'_{c0} is specified in Table 8-17. Depending on chromaEdgeFlag, the corresponding threshold variable t_{c0} is derived as follows.

- If chromaEdgeFlag is equal to 0,

$$t_{c0} = t'_{c0} * (1 << (\text{BitDepth}_Y - 8)) \quad (8-475)$$

- Otherwise (chromaEdgeFlag is equal to 1),

$$t_{c0} = t'_{c0} * (1 << (\text{BitDepth}_C - 8)) \quad (8-476)$$

Let a_p and a_q be two threshold variables specified by

$$a_p = \text{Abs}(p_2 - p_0) \quad (8-477)$$

$$a_q = \text{Abs}(q_2 - q_0) \quad (8-478)$$

The filtered result sample p'_1 is derived as follows

- If chromaEdgeFlag is equal to 0 and a_p is less than β ,

$$p'_1 = p_1 + \text{Clip3}(-t_{C0}, t_{C0}, (p_2 + ((p_0 + q_0 + 1) \gg 1) - (p_1 \ll 1)) \gg 1) \quad (8-479)$$

- Otherwise (chromaEdgeFlag is equal to 1 or a_p is greater than or equal to β),

$$p'_1 = p_1 \quad (8-480)$$

The filtered result sample q'_1 is derived as follows

- If chromaEdgeFlag is equal to 0 and a_q is less than β ,

$$q'_1 = q_1 + \text{Clip3}(-t_{C0}, t_{C0}, (q_2 + ((p_0 + q_0 + 1) \gg 1) - (q_1 \ll 1)) \gg 1) \quad (8-481)$$

- Otherwise (chromaEdgeFlag is equal to 1 or a_q is greater than or equal to β),

$$q'_1 = q_1 \quad (8-482)$$

The filtered result samples p'_2 and q'_2 are always set equal to the input samples p_2 and q_2 :

$$p'_2 = p_2 \quad (8-483)$$

$$q'_2 = q_2 \quad (8-484)$$

Table 8-17 – Value of variable t'_{C0} as a function of indexA and bS

	indexA																									
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
bS = 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
bS = 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
bS = 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1

Table 8-17 (concluded) – Value of variable t'_{C0} as a function of indexA and bS

	indexA																									
	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
bS = 1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	4	4	4	5	6	6	7	8	9	10	11	13
bS = 2	1	1	1	1	1	2	2	2	2	3	3	3	4	4	5	5	6	7	8	8	10	11	12	13	15	17
bS = 3	1	2	2	2	2	3	3	3	4	4	4	5	6	6	7	8	9	10	11	13	14	16	18	20	23	25

8.7.2.4 Filtering process for edges for bS equal to 4

Inputs to this process are the input sample values p_i and q_i ($i = 0..3$) of a single set of samples across an edge that is to be filtered, the variable `chromaEdgeFlag`, and the values of the threshold variables α and β for the set of samples, as specified in subclause 8.7.2.

Outputs of this process are the filtered result sample values p'_i and q'_i ($i = 0..2$) for the set of input sample values.

Let a_p and a_q be two threshold variables as specified in Equations 8-477 and 8-478, respectively, in subclause 8.7.2.3.

The filtered result samples p'_i ($i = 0..2$) are derived as follows.

- If `chromaEdgeFlag` is equal to 0 and the following condition holds,

$$a_p < \beta \ \&\& \ \text{Abs}(p_0 - q_0) < ((\alpha >> 2) + 2) \quad (8-485)$$

then the variables p'_0 , p'_1 , and p'_2 are derived by

$$p'_0 = (p_2 + 2*p_1 + 2*p_0 + 2*q_0 + q_1 + 4) >> 3 \quad (8-486)$$

$$p'_1 = (p_2 + p_1 + p_0 + q_0 + 2) >> 2 \quad (8-487)$$

$$p'_2 = (2*p_3 + 3*p_2 + p_1 + p_0 + q_0 + 4) >> 3 \quad (8-488)$$

- Otherwise (`chromaEdgeFlag` is equal to 1 or the condition in Equation 8-485 does not hold), the variables p'_0 , p'_1 , and p'_2 are derived by

$$p'_0 = (2*p_1 + p_0 + q_1 + 2) >> 2 \quad (8-489)$$

$$p'_1 = p_1 \quad (8-490)$$

$$p'_2 = p_2 \quad (8-491)$$

The filtered result samples q'_i ($i = 0..2$) are derived as follows.

- If `chromaEdgeFlag` is equal to 0 and the following condition holds,

$$a_q < \beta \ \&\& \ \text{Abs}(p_0 - q_0) < ((\alpha >> 2) + 2) \quad (8-492)$$

then the variables q'_0 , q'_1 , and q'_2 are derived by

$$q'_0 = (p_1 + 2*p_0 + 2*q_0 + 2*q_1 + q_2 + 4) >> 3 \quad (8-493)$$

$$q'_1 = (p_0 + q_0 + q_1 + q_2 + 2) >> 2 \quad (8-494)$$

$$q'_2 = (2*q_3 + 3*q_2 + q_1 + q_0 + p_0 + 4) >> 3 \quad (8-495)$$

- Otherwise (`chromaEdgeFlag` is equal to 1 or the condition in Equation 8-492 does not hold), the variables q'_0 , q'_1 , and q'_2 are derived by

$$q'_0 = (2*q_1 + q_0 + p_1 + 2) >> 2 \quad (8-496)$$

$$q'_1 = q_1 \quad (8-497)$$

$$q'_2 = q_2 \quad (8-498)$$

9 Parsing process

Inputs to this process are bits from the RBSP.

Outputs of this process are syntax element values.

This process is invoked when the descriptor of a syntax element in the syntax tables in subclause 7.3 is equal to ue(v), me(v), se(v), te(v) (see subclause 9.1), ce(v) (see subclause 9.2), or ae(v) (see subclause 9.3).

9.1 Parsing process for Exp-Golomb codes

This process is invoked when the descriptor of a syntax element in the syntax tables in subclause 7.3 is equal to ue(v), me(v), se(v), or te(v). For syntax elements in subclauses 7.3.4 and 7.3.5, this process is invoked only when entropy_coding_mode_flag is equal to 0.

Inputs to this process are bits from the RBSP.

Outputs of this process are syntax element values.

Syntax elements coded as ue(v), me(v), or se(v) are Exp-Golomb-coded. Syntax elements coded as te(v) are truncated Exp-Golomb-coded. The parsing process for these syntax elements begins with reading the bits starting at the current location in the bitstream up to and including the first non-zero bit, and counting the number of leading bits that are equal to 0. This process is specified as follows:

```
leadingZeroBits = -1;
for( b = 0; !b; leadingZeroBits++ )
    b = read_bits( 1 )
```

The variable codeNum is then assigned as follows:

$$\text{codeNum} = 2^{\text{leadingZeroBits}} - 1 + \text{read_bits}(\text{leadingZeroBits})$$

where the value returned from read_bits(leadingZeroBits) is interpreted as a binary representation of an unsigned integer with most significant bit written first.

Table 9-1 illustrates the structure of the Exp-Golomb code by separating the bit string into “prefix” and “suffix” bits. The “prefix” bits are those bits that are parsed in the above pseudo-code for the computation of leadingZeroBits, and are shown as either 0 or 1 in the bit string column of Table 9-1. The “suffix” bits are those bits that are parsed in the computation of codeNum and are shown as x_i in Table 9-1, with i being in the range 0 to leadingZeroBits - 1, inclusive. Each x_i can take on values 0 or 1.

Table 9-1 – Bit strings with “prefix” and “suffix” bits and assignment to codeNum ranges (informative)

Bit string form	Range of codeNum
1	0
0 1 x_0	1-2
0 0 1 $x_1 x_0$	3-6
0 0 0 1 $x_2 x_1 x_0$	7-14
0 0 0 0 1 $x_3 x_2 x_1 x_0$	15-30
0 0 0 0 0 1 $x_4 x_3 x_2 x_1 x_0$	31-62
...	...

Table 9-2 illustrates explicitly the assignment of bit strings to codeNum values.

Table 9-2 – Exp-Golomb bit strings and codeNum in explicit form and used as ue(v) (informative)

Bit string	codeNum
1	0
0 1 0	1
0 1 1	2
0 0 1 0 0	3
0 0 1 0 1	4
0 0 1 1 0	5
0 0 1 1 1	6
0 0 0 1 0 0 0	7
0 0 0 1 0 0 1	8
0 0 0 1 0 1 0	9
...	...

Depending on the descriptor, the value of a syntax element is derived as follows.

- If the syntax element is coded as ue(v), the value of the syntax element is equal to codeNum.
- Otherwise, if the syntax element is coded as se(v), the value of the syntax element is derived by invoking the mapping process for signed Exp-Golomb codes as specified in subclause 9.1.1 with codeNum as the input.
- Otherwise, if the syntax element is coded as me(v), the value of the syntax element is derived by invoking the mapping process for coded block pattern as specified in subclause 9.1.2 with codeNum as the input.
- Otherwise (the syntax element is coded as te(v)), the range of possible values for the syntax element is determined first. The range of this syntax element may be between 0 and x, with x being greater than or equal to 1 and the range is used in the derivation of the value of the syntax element value as follows
 - If x is greater than 1, codeNum and the value of the syntax element is derived in the same way as for syntax elements coded as ue(v)
 - Otherwise (x is equal to 1), the parsing process for codeNum which is equal to the value of the syntax element is given by a process equivalent to:

$$b = \text{read_bits}(1)$$

$$\text{codeNum} = !b$$

9.1.1 Mapping process for signed Exp-Golomb codes

Input to this process is codeNum as specified in subclause 9.1.

Output of this process is a value of a syntax element coded as se(v).

The syntax element is assigned to the codeNum by ordering the syntax element by its absolute value in increasing order and representing the positive value for a given absolute value with the lower codeNum. Table 9-3 provides the assignment rule.

Table 9-3 – Assignment of syntax element to codeNum for signed Exp-Golomb coded syntax elements se(v)

codeNum	syntax element value
0	0
1	1
2	−1
3	2
4	−2
5	3
6	−3
k	$(-1)^{k+1} \text{Ceil}(k \div 2)$

9.1.2 Mapping process for coded block pattern

Input to this process is codeNum as specified in subclause 9.1.

Output of this process is a value of the syntax element coded_block_pattern coded as me(v).

Table 9-4 shows the assignment of coded_block_pattern to codeNum depending on whether the macroblock prediction mode is equal to Intra_4x4, Intra_8x8 or Inter.

Table 9-4 – Assignment of codeNum to values of coded_block_pattern for macroblock prediction modes

(a) chroma_format_idc is not equal to 0

codeNum	coded_block_pattern	
	Intra_4x4, Intra_8x8	Inter
0	47	0
1	31	16
2	15	1
3	0	2
4	23	4
5	27	8
6	29	32
7	30	3
8	7	5
9	11	10
10	13	12
11	14	15
12	39	47
13	43	7
14	45	11

codeNum	coded_block_pattern	
	Intra_4x4, Intra_8x8	Inter
15	46	13
16	16	14
17	3	6
18	5	9
19	10	31
20	12	35
21	19	37
22	21	42
23	26	44
24	28	33
25	35	34
26	37	36
27	42	40
28	44	39
29	1	43
30	2	45
31	4	46
32	8	17
33	17	18
34	18	20
35	20	24
36	24	19
37	6	21
38	9	26
39	22	28
40	25	23
41	32	27
42	33	29
43	34	30
44	36	22
45	40	25
46	38	38
47	41	41

(b) chroma_format_idc is equal to 0

codeNum	coded_block_pattern	
	Intra_4x4, Intra_8x8	Inter
0	15	0
1	0	1
2	7	2
3	11	4
4	13	8
5	14	3
6	3	5
7	5	10
8	10	12
9	12	15
10	1	7
11	2	11
12	4	13
13	8	14
14	6	6
15	9	9

9.2 CAVLC parsing process for transform coefficient levels

This process is invoked when parsing syntax elements with descriptor equal to ce(v) in subclause 7.3.5.3.1 and when entropy_coding_mode_flag is equal to 0.

Inputs to this process are bits from slice data, a maximum number of non-zero transform coefficient levels maxNumCoeff, the luma block index luma4x4BlkIdx or the chroma block index chroma4x4BlkIdx of the current block of transform coefficient levels.

Output of this process is the list coeffLevel containing transform coefficient levels of the luma block with block index luma4x4BlkIdx or the chroma block with block index chroma4x4BlkIdx.

The process is specified in the following ordered steps:

1. All transform coefficient levels, with indices from 0 to maxNumCoeff - 1, in the list coeffLevel are set equal to 0.
2. The total number of non-zero transform coefficient levels TotalCoeff(coeff_token) and the number of trailing one transform coefficient levels TrailingOnes(coeff_token) are derived by parsing coeff_token (see subclause 9.2.1) as follows.
 - If the number of non-zero transform coefficient levels TotalCoeff(coeff_token) is equal to 0, the list coeffLevel containing 0 values is returned and no further step is carried out.
 - Otherwise, the following steps are carried out.
 - a. The non-zero transform coefficient levels are derived by parsing trailing_ones_sign_flag, level_prefix, and level_suffix (see subclause 9.2.2).
 - b. The runs of zero transform coefficient levels before each non-zero transform coefficient level are derived by parsing total_zeros and run_before (see subclause 9.2.3).

- c. The level and run information are combined into the list `coeffLevel` (see subclause 9.2.4).

9.2.1 Parsing process for total number of transform coefficient levels and trailing ones

Inputs to this process are bits from slice data, a maximum number of non-zero transform coefficient levels `maxNumCoeff`, the luma block index `luma4x4BlkIdx` or the chroma block index `chroma4x4BlkIdx` of the current block of transform.

Outputs of this process are `TotalCoeff(coeff_token)` and `TrailingOnes(coeff_token)`.

The syntax element `coeff_token` is decoded using one of the six VLCs specified in the six right-most columns of Table 9-5. Each VLC specifies both `TotalCoeff(coeff_token)` and `TrailingOnes(coeff_token)` for a given codeword `coeff_token`. VLC selection is dependent upon a variable `nC` that is derived as follows.

- If the CAVLC parsing process is invoked for `ChromaDCLevel`, `nC` is derived as follows.
 - If `chroma_format_idc` is equal to 1, `nC` is set equal to -1,
 - Otherwise, if `chroma_format_idc` is equal to 2, `nC` is set equal to -2,
 - Otherwise (`chroma_format_idc` is equal to 3), `nC` is set equal to 0.
- Otherwise, the following applies.
 - When the CAVLC parsing process is invoked for `Intra16x16DCLevel`, `luma4x4BlkIdx` is set equal to 0.
 - The variables `blkA` and `blkB` are derived as follows.
 - If the CAVLC parsing process is invoked for `Intra16x16DCLevel`, `Intra16x16ACLevel`, or `LumaLevel`, the process specified in subclause 6.4.8.3 is invoked with `luma4x4BlkIdx` as the input, and the output is assigned to `mbAddrA`, `mbAddrB`, `luma4x4BlkIdxA`, and `luma4x4BlkIdxB`. The 4x4 luma block specified by `mbAddrA\luma4x4BlkIdxA` is assigned to `blkA`, and the 4x4 luma block specified by `mbAddrB\luma4x4BlkIdxB` is assigned to `blkB`.
 - Otherwise (the CAVLC parsing process is invoked for `ChromaACLevel`), the process specified in subclause 6.4.8.4 is invoked with `chroma4x4BlkIdx` as input, and the output is assigned to `mbAddrA`, `mbAddrB`, `chroma4x4BlkIdxA`, and `chroma4x4BlkIdxB`. The 4x4 chroma block specified by `mbAddrA\iCbCr\chroma4x4BlkIdxA` is assigned to `blkA`, and the 4x4 chroma block specified by `mbAddrB\iCbCr\chroma4x4BlkIdxB` is assigned to `blkB`.
 - Let `nA` and `nB` be the number of non-zero transform coefficient levels (given by `TotalCoeff(coeff_token)`) in the block of transform coefficient levels `blkA` located to the left of the current block and the block of transform coefficient levels `blkB` located above the current block, respectively.
 - With `N` replaced by `A` and `B`, in `mbAddrN`, `blkN`, and `nN` the following applies.
 - If any of the following conditions is true, `nN` is set equal to 0.
 - `mbAddrN` is not available
 - The current macroblock is coded using an Intra prediction mode, `constrained_intra_pred_flag` is equal to 1 and `mbAddrN` is coded using Inter prediction and slice data partitioning is in use (`nal_unit_type` is in the range of 2 to 4, inclusive).
 - The macroblock `mbAddrN` has `mb_type` equal to `P_Skip` or `B_Skip`
 - All AC residual transform coefficient levels of the neighbouring block `blkN` are equal to 0 due to the corresponding bit of `CodedBlockPatternLuma` or `CodedBlockPatternChroma` being equal to 0
 - Otherwise, if `mbAddrN` is an `I_PCM` macroblock, `nN` is set equal to 16.
 - Otherwise, `nN` is set equal to the value `TotalCoeff(coeff_token)` of the neighbouring block `blkN`.

NOTE 1 – The values `nA` and `nB` that are derived using `TotalCoeff(coeff_token)` do not include the DC transform coefficient levels in `Intra_16x16` macroblocks or DC transform coefficient levels in chroma blocks, because these transform coefficient levels are decoded separately. When the block above or to the left belongs to an `Intra_16x16` macroblock, or is a chroma block, `nA` and `nB` is the number of decoded non-zero AC transform coefficient levels.

NOTE 2 – When parsing for `Intra16x16DCLevel`, the values `nA` and `nB` are based on the number of non-zero transform coefficient levels in adjacent 4x4 blocks and not on the number of non-zero DC transform coefficient levels in adjacent 16x16 blocks.
- Given the values of `nA` and `nB`, the variable `nC` is derived as follows.

- If both mbAddrA and mbAddrB are available, the variable nC is set equal to $(nA + nB + 1) >> 1$.
- Otherwise (mbAddrA is not available or mbAddrB is not available), the variable nC is set equal to nA + nB.

The value of TotalCoeff(coeff_token) resulting from decoding coeff_token shall be in the range of 0 to maxNumCoeff, inclusive.

Table 9-5 – coeff_token mapping to TotalCoeff(coeff_token) and TrailingOnes(coeff_token)

TrailingOnes (coeff_token)	TotalCoeff (coeff_token)	$0 \leq nC < 2$	$2 \leq nC < 4$	$4 \leq nC < 8$	$8 \leq nC$	$nC = -1$	$nC = -2$
0	0	1	11	1111	0000 11	01	1
0	1	0001 01	0010 11	0011 11	0000 00	0001 11	0001 111
1	1	01	10	1110	0000 01	1	01
0	2	0000 0111	0001 11	0010 11	0001 00	0001 00	0001 110
1	2	0001 00	0011 1	0111 1	0001 01	0001 10	0001 101
2	2	001	011	1101	0001 10	001	001
0	3	0000 0011 1	0000 111	0010 00	0010 00	0000 11	0000 0011 1
1	3	0000 0110	0010 10	0110 0	0010 01	0000 011	0001 100
2	3	0000 101	0010 01	0111 0	0010 10	0000 010	0001 011
3	3	0001 1	0101	1100	0010 11	0001 01	0000 1
0	4	0000 0001 11	0000 0111	0001 111	0011 00	0000 10	0000 0011 0
1	4	0000 0011 0	0001 10	0101 0	0011 01	0000 0011	0000 0010 1
2	4	0000 0101	0001 01	0101 1	0011 10	0000 0010	0001 010
3	4	0000 11	0100	1011	0011 11	0000 000	0000 01
0	5	0000 0000 111	0000 0100	0001 011	0100 00	-	0000 0001 11
1	5	0000 0001 10	0000 110	0100 0	0100 01	-	0000 0001 10
2	5	0000 0010 1	0000 101	0100 1	0100 10	-	0000 0010 0
3	5	0000 100	0011 0	1010	0100 11	-	0001 001
0	6	0000 0000 0111 1	0000 0011 1	0001 001	0101 00	-	0000 0000 111
1	6	0000 0000 110	0000 0110	0011 10	0101 01	-	0000 0000 110
2	6	0000 0001 01	0000 0101	0011 01	0101 10	-	0000 0001 01
3	6	0000 0100	0010 00	1001	0101 11	-	0001 000
0	7	0000 0000 0101 1	0000 0001 111	0001 000	0110 00	-	0000 0000 0111
1	7	0000 0000 0111 0	0000 0011 0	0010 10	0110 01	-	0000 0000 0110
2	7	0000 0000 101	0000 0010 1	0010 01	0110 10	-	0000 0000 101
3	7	0000 0010 0	0001 00	1000	0110 11	-	0000 0001 00
0	8	0000 0000 0100 0	0000 0001 011	0000 1111	0111 00	-	0000 0000 0011 1
1	8	0000 0000 0101 0	0000 0001 110	0001 110	0111 01	-	0000 0000 0101
2	8	0000 0000 0110 1	0000 0001 101	0001 101	0111 10	-	0000 0000 0100
3	8	0000 0001 00	0000 100	0110 1	0111 11	-	0000 0000 100
0	9	0000 0000 0011 11	0000 0000 1111	0000 1011	1000 00	-	

TrailingOnes (coeff_token)	TotalCoeff (coeff_token)	$0 \leq nC < 2$	$2 \leq nC < 4$	$4 \leq nC < 8$	$8 \leq nC$	$nC == -1$	$nC == -2$
1	9	0000 0000 0011 10	0000 0001 010	0000 1110	1000 01	-	
2	9	0000 0000 0100 1	0000 0001 001	0001 010	1000 10	-	
3	9	0000 0000 100	0000 0010 0	0011 00	1000 11	-	
0	10	0000 0000 0010 11	0000 0000 1011	0000 0111 1	1001 00	-	
1	10	0000 0000 0010 10	0000 0000 1110	0000 1010	1001 01	-	
2	10	0000 0000 0011 01	0000 0000 1101	0000 1101	1001 10	-	
3	10	0000 0000 0110 0	0000 0001 100	0001 100	1001 11	-	
0	11	0000 0000 0001 111	0000 0000 1000	0000 0101 1	1010 00	-	
1	11	0000 0000 0001 110	0000 0000 1010	0000 0111 0	1010 01	-	
2	11	0000 0000 0010 01	0000 0000 1001	0000 1001	1010 10	-	
3	11	0000 0000 0011 00	0000 0001 000	0000 1100	1010 11	-	
0	12	0000 0000 0001 011	0000 0000 0111 1	0000 0100 0	1011 00	-	
1	12	0000 0000 0001 010	0000 0000 0111 0	0000 0101 0	1011 01	-	
2	12	0000 0000 0001 101	0000 0000 0110 1	0000 0110 1	1011 10	-	
3	12	0000 0000 0010 00	0000 0000 1100	0000 1000	1011 11	-	
0	13	0000 0000 0000 1111	0000 0000 0101 1	0000 0011 01	1100 00	-	
1	13	0000 0000 0000 001	0000 0000 0101 0	0000 0011 1	1100 01	-	
2	13	0000 0000 0001 001	0000 0000 0100 1	0000 0100 1	1100 10	-	
3	13	0000 0000 0001 100	0000 0000 0110 0	0000 0110 0	1100 11	-	
0	14	0000 0000 0000 1011	0000 0000 0011 1	0000 0010 01	1101 00	-	
1	14	0000 0000 0000 1110	0000 0000 0010 11	0000 0011 00	1101 01	-	
2	14	0000 0000 0000 1101	0000 0000 0011 0	0000 0010 11	1101 10	-	
3	14	0000 0000 0001 000	0000 0000 0100 0	0000 0010 10	1101 11	-	
0	15	0000 0000 0000 0111	0000 0000 0010 01	0000 0001 01	1110 00	-	
1	15	0000 0000 0000 1010	0000 0000 0010 00	0000 0010 00	1110 01	-	
2	15	0000 0000 0000 1001	0000 0000 0010 10	0000 0001 11	1110 10	-	
3	15	0000 0000 0000 1100	0000 0000 0000 1	0000 0001 10	1110 11	-	
0	16	0000 0000 0000 0100	0000 0000 0001 11	0000 0000 01	1111 00	-	
1	16	0000 0000 0000 0110	0000 0000 0001 10	0000 0001 00	1111 01	-	
2	16	0000 0000 0000 0101	0000 0000 0001 01	0000 0000 11	1111 10	-	
3	16	0000 0000 0000 1000	0000 0000 0001 00	0000 0000 10	1111 11	-	

9.2.2 Parsing process for level information

Inputs to this process are bits from slice data, the number of non-zero transform coefficient levels TotalCoeff(coeff_token), and the number of trailing one transform coefficient levels TrailingOnes(coeff_token).

Output of this process is a list with name level containing transform coefficient levels.

Initially an index i is set equal to 0. Then the following procedure is iteratively applied $\text{TrailingOnes}(\text{coeff_token})$ times to decode the trailing one transform coefficient levels (if any):

- A 1-bit syntax element $\text{trailing_ones_sign_flag}$ is decoded and evaluated as follows.
 - If $\text{trailing_ones_sign_flag}$ is equal to 0, the value +1 is assigned to $\text{level}[i]$.
 - Otherwise ($\text{trailing_ones_sign_flag}$ is equal to 1), the value -1 is assigned to $\text{level}[i]$.
- The index i is incremented by 1.

Following the decoding of the trailing one transform coefficient levels, a variable suffixLength is initialised as follows.

- If $\text{TotalCoeff}(\text{coeff_token})$ is greater than 10 and $\text{TrailingOnes}(\text{coeff_token})$ is less than 3, suffixLength is set equal to 1.
- Otherwise ($\text{TotalCoeff}(\text{coeff_token})$ is less than or equal to 10 or $\text{TrailingOnes}(\text{coeff_token})$ is equal to 3), suffixLength is set equal to 0.

The following procedure is then applied iteratively ($\text{TotalCoeff}(\text{coeff_token}) - \text{TrailingOnes}(\text{coeff_token})$) times to decode the remaining levels (if any):

- The syntax element level_prefix is decoded as specified in subclause 9.2.2.1.
- The variable levelSuffixSize is set equal to the variable suffixLength with the exception of the following two cases.
- When level_prefix is equal to 14 and suffixLength is equal to 0, levelSuffixSize is set equal to 4.
- When level_prefix is greater than or equal to 15, levelSuffixSize is set equal to $\text{level_prefix} - 3$.
- The syntax element level_suffix is decoded as follows.
 - If levelSuffixSize is greater than 0, the syntax element level_suffix is decoded as unsigned integer representation $u(v)$ with levelSuffixSize bits.
 - Otherwise (levelSuffixSize is equal to 0), the syntax element level_suffix is inferred to be equal to 0.
- A variable levelCode is set equal to $(\text{Min}(15, \text{level_prefix}) \ll \text{suffixLength}) + \text{level_suffix}$.
- When level_prefix is greater than or equal to 15 and suffixLength is equal to 0, levelCode is incremented by 15.
- When level_prefix is greater than or equal to 16, levelCode is incremented by $(1 \ll (\text{level_prefix} - 3)) - 4096$.
- When the index i is equal to $\text{TrailingOnes}(\text{coeff_token})$ and $\text{TrailingOnes}(\text{coeff_token})$ is less than 3, levelCode is incremented by 2.
- The variable $\text{level}[i]$ is derived as follows.
 - If levelCode is an even number, the value $(\text{levelCode} + 2) \gg 1$ is assigned to $\text{level}[i]$.
 - Otherwise (levelCode is an odd number), the value $(-\text{levelCode} - 1) \gg 1$ is assigned to $\text{level}[i]$.
- When suffixLength is equal to 0, suffixLength is set equal to 1.
- When the absolute value of $\text{level}[i]$ is greater than $(3 \ll (\text{suffixLength} - 1))$ and suffixLength is less than 6, suffixLength is incremented by 1.
- The index i is incremented by 1.

9.2.2.1 Parsing process for level_prefix

Inputs to this process are bits from slice data.

Output of this process is level_prefix .

The parsing process for this syntax element consists in reading the bits starting at the current location in the bitstream up to and including the first non-zero bit, and counting the number of leading bits that are equal to 0. This process is specified as follows:

```

leadingZeroBits = -1
for( b = 0; !b; leadingZeroBits++ )
    b = read_bits( 1 )
level_prefix = leadingZeroBits

```

Table 9-6 illustrates the codeword table for level_prefix.

Table 9-6 – Codeword table for level_prefix (informative)

level_prefix	bit string
0	1
1	01
2	001
3	0001
4	0000 1
5	0000 01
6	0000 001
7	0000 0001
8	0000 0000 1
9	0000 0000 01
10	0000 0000 001
11	0000 0000 0001
12	0000 0000 0000 1
13	0000 0000 0000 01
14	0000 0000 0000 001
15	0000 0000 0000 0001
...	...

9.2.3 Parsing process for run information

Inputs to this process are bits from slice data, the number of non-zero transform coefficient levels TotalCoeff(coeff_token), and the maximum number of non-zero transform coefficient levels maxNumCoeff.

Output of this process is a list of runs of zero transform coefficient levels preceding non-zero transform coefficient levels called run.

Initially, an index i is set equal to 0.

The variable zerosLeft is derived as follows.

- If the number of non-zero transform coefficient levels TotalCoeff(coeff_token) is equal to the maximum number of non-zero transform coefficient levels maxNumCoeff, a variable zerosLeft is set equal to 0.
- Otherwise (the number of non-zero transform coefficient levels TotalCoeff(coeff_token) is less than the maximum number of non-zero transform coefficient levels maxNumCoeff), total_zeros is decoded and zerosLeft is set equal to its value.

The VLC used to decode total_zeros is derived as follows:

If maxNumCoeff is equal to 4, one of the VLCs specified in Table 9-9 (a) is used.

- Otherwise, if maxNumCoeff is equal to 8, one of the VLCs specified in Table 9-9 (b) is used.
- Otherwise (maxNumCoeff is not equal to 4 and not equal to 8), VLCs from Tables 9-7 and 9-8 are used.

The following procedure is then applied iteratively (TotalCoeff(coeff_token) – 1) times:

- The variable run[i] is derived as follows.
 - If zerosLeft is greater than zero, a value run_before is decoded based on Table 9-10 and zerosLeft. run[i] is set equal to run_before.
 - Otherwise (zerosLeft is equal to 0), run[i] is set equal to 0.
- The value of run[i] is subtracted from zerosLeft and the result assigned to zerosLeft. The result of the subtraction shall be greater than or equal to 0.
- The index i is incremented by 1.

Finally the value of zerosLeft is assigned to run[i].

Table 9-7 – total_zeros tables for 4x4 blocks with TotalCoeff(coeff_token) 1 to 7

total_zeros	TotalCoeff(coeff_token)						
	1	2	3	4	5	6	7
0	1	111	0101	0001 1	0101	0000 01	0000 01
1	011	110	111	111	0100	0000 1	0000 1
2	010	101	110	0101	0011	111	101
3	0011	100	101	0100	111	110	100
4	0010	011	0100	110	110	101	011
5	0001 1	0101	0011	101	101	100	11
6	0001 0	0100	100	100	100	011	010
7	0000 11	0011	011	0011	011	010	0001
8	0000 10	0010	0010	011	0010	0001	001
9	0000 011	0001 1	0001 1	0010	0000 1	001	0000 00
10	0000 010	0001 0	0001 0	0001 0	0001	0000 00	
11	0000 0011	0000 11	0000 01	0000 1	0000 0		
12	0000 0010	0000 10	0000 1	0000 0			
13	0000 0001 1	0000 01	0000 00				
14	0000 0001 0	0000 00					
15	0000 0000 1						

Table 9-8 – total_zeros tables for 4x4 blocks with TotalCoeff(coeff_token) 8 to 15

total_zeros	TotalCoeff(coeff_token)							
	8	9	10	11	12	13	14	15
0	0000 01	0000 01	0000 1	0000	0000	000	00	0
1	0001	0000 00	0000 0	0001	0001	001	01	1
2	0000 1	0001	001	001	01	1	1	
3	011	11	11	010	1	01		
4	11	10	10	1	001			
5	10	001	01	011				
6	010	01	0001					
7	001	0000 1						
8	0000 00							

Table 9-9 – total_zeros tables for chroma DC 2x2 and 2x4 blocks

(a) Chroma DC 2x2 block (4:2:0 chroma sampling)

total_zeros	TotalCoeff(coeff_token)		
	1	2	3
0	1	1	1
1	01	01	0
2	001	00	
3	000		

(b) Chroma DC 2x4 block (4:2:2 chroma sampling)

total_zeros	TotalCoeff(coeff_token)						
	1	2	3	4	5	6	7
0	1	000	000	110	00	00	0
1	010	01	001	00	01	01	1
2	011	001	01	01	10	1	
3	0010	100	10	10	11		
4	0011	101	110	111			
5	0001	110	111				
6	0000 1	111					
7	0000 0						

Table 9-10 – Tables for run_before

run_before	zerosLeft						
	1	2	3	4	5	6	>6
0	1	1	11	11	11	11	111
1	0	01	10	10	10	000	110
2	-	00	01	01	011	001	101
3	-	-	00	001	010	011	100
4	-	-	-	000	001	010	011
5	-	-	-	-	000	101	010
6	-	-	-	-	-	100	001
7	-	-	-	-	-	-	0001
8	-	-	-	-	-	-	00001
9	-	-	-	-	-	-	000001
10	-	-	-	-	-	-	0000001
11	-	-	-	-	-	-	00000001
12	-	-	-	-	-	-	000000001
13	-	-	-	-	-	-	0000000001
14	-	-	-	-	-	-	00000000001

9.2.4 Combining level and run information

Input to this process are a list of transform coefficient levels called level, a list of runs called run, and the number of non-zero transform coefficient levels TotalCoeff(coeff_token).

Output of this process is an list coeffLevel of transform coefficient levels.

A variable coeffNum is set equal to -1 and an index i is set equal to (TotalCoeff(coeff_token) – 1). The following procedure is iteratively applied TotalCoeff(coeff_token) times:

- coeffNum is incremented by run[i] + 1.
- coeffLevel[coeffNum] is set equal to level[i].
- The index i is decremented by 1.

9.3 CABAC parsing process for slice data

This process is invoked when parsing syntax elements with descriptor ae(v) in subclauses 7.3.4 and 7.3.5 when entropy_coding_mode_flag is equal to 1.

Inputs to this process are a request for a value of a syntax element and values of prior parsed syntax elements.

Output of this process is the value of the syntax element.

When starting the parsing of the slice data of a slice in subclause 7.3.4, the initialisation process of the CABAC parsing process is invoked as specified in subclause 9.3.1.

The parsing of syntax elements proceeds as follows:

For each requested value of a syntax element a binarization is derived as described in subclause 9.3.2.

The binarization for the syntax element and the sequence of parsed bins determines the decoding process flow as described in subclause 9.3.3.

For each bin of the binarization of the syntax element, which is indexed by the variable `binIdx`, a context index `ctxIdx` is derived as specified in subclause 9.3.3.1.

For each `ctxIdx` the arithmetic decoding process is invoked as specified in subclause 9.3.3.2.

The resulting sequence ($b_0 \dots b_{\text{binIdx}}$) of parsed bins is compared to the set of bin strings given by the binarization process after decoding of each bin. When the sequence matches a bin string in the given set, the corresponding value is assigned to the syntax element.

In case the request for a value of a syntax element is processed for the syntax element `mb_type` and the decoded value of `mb_type` is equal to `I_PCM`, the decoding engine is initialised after the decoding of any `pcm_alignment_zero_bit` and all `pcm_sample_luma` and `pcm_sample_chroma` data as specified in subclause 9.3.1.2.

The whole CABAC parsing process is illustrated in the flowchart of Figure 9-1 with the abbreviation SE for syntax element.

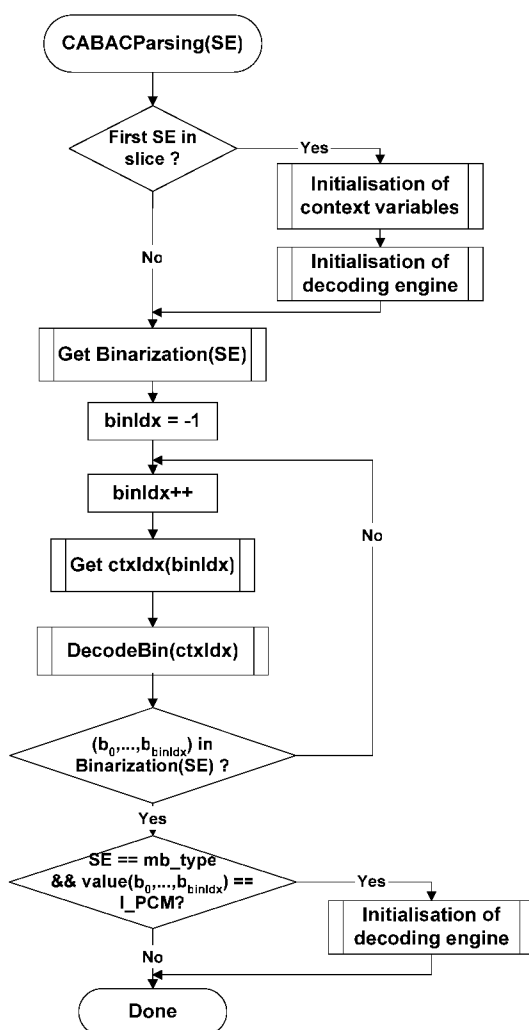


Figure 9-1 – Illustration of CABAC parsing process for a syntax element SE (informative)

9.3.1 Initialisation process

Outputs of this process are initialised CABAC internal variables.

The processes in subclauses 9.3.1.1 and 9.3.1.2 are invoked when starting the parsing of the slice data of a slice in subclause 7.3.4.

The process in subclause 9.3.1.2 is also invoked after decoding any pcm_alignment_zero_bit and all pcm_sample_luma and pcm_sample_chroma data for a macroblock of type I_PCM.

9.3.1.1 Initialisation process for context variables

Outputs of this process are the initialised CABAC context variables indexed by ctxIdx.

Table 9-12 to Table 9-23 contain the values of the variables n and m used in the initialisation of context variables that are assigned to all syntax elements in subclauses 7.3.4 and 7.3.5 except for the end-of-slice flag.

For each context variable, the two variables pStateIdx and valMPS are initialised.

NOTE 1 – The variable pStateIdx corresponds to a probability state index and the variable valMPS corresponds to the value of the most probable symbol as further described in subclause 9.3.3.2.

The two values assigned to pStateIdx and valMPS for the initialisation are derived from SliceQP_Y, which is derived in Equation 7-27. Given the two table entries (m, n),

1. $\text{preCtxState} = \text{Clip3}(1, 126, ((m * \text{Clip3}(0, 51, \text{SliceQP}_Y)) \gg 4) + n)$
2. $\text{if}(\text{preCtxState} \leq 63) \{$
 $\quad \text{pStateIdx} = 63 - \text{preCtxState}$
 $\quad \text{valMPS} = 0$
 $\quad \text{else} \{$
 $\quad \quad \text{pStateIdx} = \text{preCtxState} - 64$
 $\quad \quad \text{valMPS} = 1$
 $\quad \}$
 $\}$

In Table 9-11, the ctxIdx for which initialisation is needed for each of the slice types are listed. Also listed is the table number that includes the values of m and n needed for the initialisation. For P, SP and B slice type, the initialisation depends also on the value of the cabac_init_idc syntax element. Note that the syntax element names do not affect the initialisation process.

Table 9-11 – Association of ctxIdx and syntax elements for each slice type in the initialisation process

	Syntax element	Table	Slice type			
			SI	I	P, SP	B
slice_data()	mb_skip_flag	Table 9-13 Table 9-14			11-13	24-26
	mb_field_decoding_flag	Table 9-18	70-72	70-72	70-72	70-72
macroblock_layer()	mb_type	Table 9-12 Table 9-13 Table 9-14	0-10	3-10	14-20	27-35
	transform_size_8x8_flag	Table 9-16	na	399-401	399-401	399-401
	coded_block_pattern (luma)	Table 9-18	73-76	73-76	73-76	73-76
	coded_block_pattern (chroma)	Table 9-18	77-84	77-84	77-84	77-84
	mb_qp_delta	Table 9-17	60-63	60-63	60-63	60-63
mb_pred()	prev_intra4x4_pred_mode_flag	Table 9-17	68	68	68	68
	rem_intra4x4_pred_mode	Table 9-17	69	69	69	69
	prev_intra8x8_pred_mode_flag	Table 9-17	na	68	68	68
	rem_intra8x8_pred_mode	Table 9-17	na	69	69	69
	intra_chroma_pred_mode	Table 9-17	64-67	64-67	64-67	64-67

Table 9-11 – Association of ctxIdx and syntax elements for each slice type in the initialisation process

	Syntax element	Table	Slice type			
			SI	I	P, SP	B
mb_pred() and sub_mb_pred()	ref_idx_l0	Table 9-16			54-59	54-59
	ref_idx_l1	Table 9-16				54-59
	mvd_l0[][][0]	Table 9-15			40-46	40-46
	mvd_l1[][][0]	Table 9-15				40-46
	mvd_l0[][][1]	Table 9-15			47-53	47-53
	mvd_l1[][][1]	Table 9-15				47-53
sub_mb_pred()	sub_mb_type	Table 9-13			21-23	36-39
		Table 9-14				
residual_block_cabac()	coded_block_flag	Table 9-18	85-104	85-104	85-104	85-104
	significant_coeff_flag[]	Table 9-19	105-165 277-337	105-165 277-337 402-416 436-450	105-165 277-337 402-416 436-450	105-165 277-337 402-416 436-450
		Table 9-22				
		Table 9-24				
	last_significant_coeff_flag[]	Table 9-20	166-226 338-398	166-226 338-398 417-425 451-459	166-226 338-398 417-425 451-459	166-226 338-398 417-425 451-459
		Table 9-23				
		Table 9-24				
	coeff_abs_level_minus1[]	Table 9-21 Table 9-24	227-275	227-275 426-435	227-275 426-435	227-275 426-435

NOTE 2 – ctxIdx equal to 276 is associated with the end_of_slice_flag and the bin of mb_type, which specifies the I_PCM macroblock type. The decoding process specified in subclause 9.3.3.2.4 applies to ctxIdx equal to 276. This decoding process, however, may also be implemented by using the decoding process specified in subclause 9.3.3.2.1. In this case, the initial values associated with ctxIdx equal to 276 are specified to be pStateIdx = 63 and valMPS = 0, where pStateIdx = 63 represents a non-adapting probability state.

Table 9-12 – Values of variables m and n for ctxIdx from 0 to 10

Initialisation variables	ctxIdx										
	0	1	2	3	4	5	6	7	8	9	10
m	20	2	3	20	2	3	-28	-23	-6	-1	7
n	-15	54	74	-15	54	74	127	104	53	54	51

Table 9-13 – Values of variables m and n for ctxIdx from 11 to 23

Value of cabac_init_idc	Initialisation variables	ctxIdx												
		11	12	13	14	15	16	17	18	19	20	21	22	23
0	m	23	23	21	1	0	-37	5	-13	-11	1	12	-4	17
	n	33	2	0	9	49	118	57	78	65	62	49	73	50
1	m	22	34	16	-2	4	-29	2	-6	-13	5	9	-3	10
	n	25	0	0	9	41	118	65	71	79	52	50	70	54
2	m	29	25	14	-10	-3	-27	26	-4	-24	5	6	-17	14
	n	16	0	0	51	62	99	16	85	102	57	57	73	57

Table 9-14 – Values of variables m and n for ctxIdx from 24 to 39

Value of cabac_init_idc	Initialisation variables	ctxIdx															
		24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
0	m	18	9	29	26	16	9	-46	-20	1	-13	-11	1	-6	-17	-6	9
	n	64	43	0	67	90	104	127	104	67	78	65	62	86	95	61	45
1	m	26	19	40	57	41	26	-45	-15	-4	-6	-13	5	6	-13	0	8
	n	34	22	0	2	36	69	127	101	76	71	79	52	69	90	52	43
2	m	20	20	29	54	37	12	-32	-22	-2	-4	-24	5	-6	-14	-6	4
	n	40	10	0	0	42	97	127	117	74	85	102	57	93	88	44	55

Table 9-15 – Values of variables m and n for ctxIdx from 40 to 53

Value of cabac_init_idc	Initialisation variables	ctxIdx													
		40	41	42	43	44	45	46	47	48	49	50	51	52	53
0	m	-3	-6	-11	6	7	-5	2	0	-3	-10	5	4	-3	0
	n	69	81	96	55	67	86	88	58	76	94	54	69	81	88
1	m	-2	-5	-10	2	2	-3	-3	1	-3	-6	0	-3	-7	-5
	n	69	82	96	59	75	87	100	56	74	85	59	81	86	95
2	m	-11	-15	-21	19	20	4	6	1	-5	-13	5	6	-3	-1
	n	89	103	116	57	58	84	96	63	85	106	63	75	90	101

Table 9-16 – Values of variables m and n for ctxIdx from 54 to 59, and 399 to 401

Value of cabac_init_idc	Initialisation variables	ctxIdx								
		54	55	56	57	58	59	399	400	401
I slices	m	na	na	na	na	na	na	31	31	25
	n	na	na	na	na	na	na	21	31	50
0	m	-7	-5	-4	-5	-7	1	12	11	14
	n	67	74	74	80	72	58	40	51	59
1	m	-1	-1	1	-2	-5	0	25	21	21
	n	66	77	70	86	72	61	32	49	54
2	m	3	-4	-2	-12	-7	1	21	19	17
	n	55	79	75	97	50	60	33	50	61

Table 9-17 – Values of variables m and n for ctxIdx from 60 to 69

Initialisation variables	ctxIdx									
	60	61	62	63	64	65	66	67	68	69
m	0	0	0	0	-9	4	0	-7	13	3
n	41	63	63	63	83	86	97	72	41	62

Table 9-18 – Values of variables m and n for ctxIdx from 70 to 104

ctxIdx	I and SI slices		Value of cabac_init_idc						ctxIdx	I and SI slices		Value of cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
70	0	11	0	45	13	15	7	34	88	-11	115	-13	108	-4	92	5	78
71	1	55	-4	78	7	51	-9	88	89	-12	63	-3	46	0	39	-6	55
72	0	69	-3	96	2	80	-20	127	90	-2	68	-1	65	0	65	4	61
73	-17	127	-27	126	-39	127	-36	127	91	-15	84	-1	57	-15	84	-14	83
74	-13	102	-28	98	-18	91	-17	91	92	-13	104	-9	93	-35	127	-37	127
75	0	82	-25	101	-17	96	-14	95	93	-3	70	-3	74	-2	73	-5	79
76	-7	74	-23	67	-26	81	-25	84	94	-8	93	-9	92	-12	104	-11	104
77	-21	107	-28	82	-35	98	-25	86	95	-10	90	-8	87	-9	91	-11	91
78	-27	127	-20	94	-24	102	-12	89	96	-30	127	-23	126	-31	127	-30	127
79	-31	127	-16	83	-23	97	-17	91	97	-1	74	5	54	3	55	0	65
80	-24	127	-22	110	-27	119	-31	127	98	-6	97	6	60	7	56	-2	79
81	-18	95	-21	91	-24	99	-14	76	99	-7	91	6	59	7	55	0	72
82	-27	127	-18	102	-21	110	-18	103	100	-20	127	6	69	8	61	-4	92
83	-21	114	-13	93	-18	102	-13	90	101	-4	56	-1	48	-3	53	-6	56
84	-30	127	-29	127	-36	127	-37	127	102	-5	82	0	68	0	68	3	68
85	-17	123	-7	92	0	80	11	80	103	-7	76	-4	69	-7	74	-8	71
86	-12	115	-5	89	-5	89	5	76	104	-22	125	-8	88	-9	88	-13	98
87	-16	122	-7	96	-7	94	2	84									

Table 9-19 – Values of variables m and n for ctxIdx from 105 to 165

ctxIdx	I and SI slices		Value of cabac_init_idc						ctxIdx	I and SI slices		Value of cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
105	-7	93	-2	85	-13	103	-4	86	136	-13	101	5	53	0	58	-5	75
106	-11	87	-6	78	-13	91	-12	88	137	-13	91	-2	61	-1	60	-8	80
107	-3	77	-1	75	-9	89	-5	82	138	-12	94	0	56	-3	61	-21	83
108	-5	71	-7	77	-14	92	-3	72	139	-10	88	0	56	-8	67	-21	64
109	-4	63	2	54	-8	76	-4	67	140	-16	84	-13	63	-25	84	-13	31
110	-4	68	5	50	-12	87	-8	72	141	-10	86	-5	60	-14	74	-25	64
111	-12	84	-3	68	-23	110	-16	89	142	-7	83	-1	62	-5	65	-29	94
112	-7	62	1	50	-24	105	-9	69	143	-13	87	4	57	5	52	9	75
113	-7	65	6	42	-10	78	-1	59	144	-19	94	-6	69	2	57	17	63
114	8	61	-4	81	-20	112	5	66	145	1	70	4	57	0	61	-8	74
115	5	56	1	63	-17	99	4	57	146	0	72	14	39	-9	69	-5	35
116	-2	66	-4	70	-78	127	-4	71	147	-5	74	4	51	-11	70	-2	27
117	1	64	0	67	-70	127	-2	71	148	18	59	13	68	18	55	13	91
118	0	61	2	57	-50	127	2	58	149	-8	102	3	64	-4	71	3	65
119	-2	78	-2	76	-46	127	-1	74	150	-15	100	1	61	0	58	-7	69
120	1	50	11	35	-4	66	-4	44	151	0	95	9	63	7	61	8	77
121	7	52	4	64	-5	78	-1	69	152	-4	75	7	50	9	41	-10	66
122	10	35	1	61	-4	71	0	62	153	2	72	16	39	18	25	3	62
123	0	44	11	35	-8	72	-7	51	154	-11	75	5	44	9	32	-3	68
124	11	38	18	25	2	59	-4	47	155	-3	71	4	52	5	43	-20	81
125	1	45	12	24	-1	55	-6	42	156	15	46	11	48	9	47	0	30
126	0	46	13	29	-7	70	-3	41	157	-13	69	-5	60	0	44	1	7
127	5	44	13	36	-6	75	-6	53	158	0	62	-1	59	0	51	-3	23
128	31	17	-10	93	-8	89	8	76	159	0	65	0	59	2	46	-21	74
129	1	51	-7	73	-34	119	-9	78	160	21	37	22	33	19	38	16	66
130	7	50	-2	73	-3	75	-11	83	161	-15	72	5	44	-4	66	-23	124
131	28	19	13	46	32	20	9	52	162	9	57	14	43	15	38	17	37
132	16	33	9	49	30	22	0	67	163	16	54	-1	78	12	42	44	-18
133	14	62	-7	100	-44	127	-5	90	164	0	62	0	60	9	34	50	-34
134	-13	108	9	53	0	54	1	67	165	12	72	9	69	0	89	-22	127
135	-15	100	2	53	-5	61	-15	72									

Table 9-20 – Values of variables m and n for ctxIdx from 166 to 226

ctxIdx	I and SI slices		Value of cabac_init_idc						ctxIdx	I and SI slices		Value of cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
166	24	0	11	28	4	45	4	39	197	26	-17	28	3	36	-28	28	-3
167	15	9	2	40	10	28	0	42	198	30	-25	28	4	38	-28	24	10
168	8	25	3	44	10	31	7	34	199	28	-20	32	0	38	-27	27	0
169	13	18	0	49	33	-11	11	29	200	33	-23	34	-1	34	-18	34	-14
170	15	9	0	46	52	-43	8	31	201	37	-27	30	6	35	-16	52	-44
171	13	19	2	44	18	15	6	37	202	33	-23	30	6	34	-14	39	-24
172	10	37	2	51	28	0	7	42	203	40	-28	32	9	32	-8	19	17
173	12	18	0	47	35	-22	3	40	204	38	-17	31	19	37	-6	31	25
174	6	29	4	39	38	-25	8	33	205	33	-11	26	27	35	0	36	29
175	20	33	2	62	34	0	13	43	206	40	-15	26	30	30	10	24	33
176	15	30	6	46	39	-18	13	36	207	41	-6	37	20	28	18	34	15
177	4	45	0	54	32	-12	4	47	208	38	1	28	34	26	25	30	20
178	1	58	3	54	102	-94	3	55	209	41	17	17	70	29	41	22	73
179	0	62	2	58	0	0	2	58	210	30	-6	1	67	0	75	20	34
180	7	61	4	63	56	-15	6	60	211	27	3	5	59	2	72	19	31
181	12	38	6	51	33	-4	8	44	212	26	22	9	67	8	77	27	44
182	11	45	6	57	29	10	11	44	213	37	-16	16	30	14	35	19	16
183	15	39	7	53	37	-5	14	42	214	35	-4	18	32	18	31	15	36
184	11	42	6	52	51	-29	7	48	215	38	-8	18	35	17	35	15	36
185	13	44	6	55	39	-9	4	56	216	38	-3	22	29	21	30	21	28
186	16	45	11	45	52	-34	4	52	217	37	3	24	31	17	45	25	21
187	12	41	14	36	69	-58	13	37	218	38	5	23	38	20	42	30	20
188	10	49	8	53	67	-63	9	49	219	42	0	18	43	18	45	31	12
189	30	34	-1	82	44	-5	19	58	220	35	16	20	41	27	26	27	16
190	18	42	7	55	32	7	10	48	221	39	22	11	63	16	54	24	42
191	10	55	-3	78	55	-29	12	45	222	14	48	9	59	7	66	0	93
192	17	51	15	46	32	1	0	69	223	27	37	9	64	16	56	14	56
193	17	46	22	31	0	0	20	33	224	21	60	-1	94	11	73	15	57
194	0	89	-1	84	27	36	8	63	225	12	68	-2	89	10	67	26	38
195	26	-19	25	7	33	-25	35	-18	226	2	97	-9	108	-10	116	-24	127
196	22	-17	30	-7	34	-30	33	-25									

Table 9-21 – Values of variables m and n for ctxIdx from 227 to 275

ctxIdx	I and SI slices		Value of cabac_init_idc						ctxIdx	I and SI slices		Value of cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
227	-3	71	-6	76	-23	112	-24	115	252	-12	73	-6	55	-16	72	-14	75
228	-6	42	-2	44	-15	71	-22	82	253	-8	76	0	58	-7	69	-10	79
229	-5	50	0	45	-7	61	-9	62	254	-7	80	0	64	-4	69	-9	83
230	-3	54	0	52	0	53	0	53	255	-9	88	-3	74	-5	74	-12	92
231	-2	62	-3	64	-5	66	0	59	256	-17	110	-10	90	-9	86	-18	108
232	0	58	-2	59	-11	77	-14	85	257	-11	97	0	70	2	66	-4	79
233	1	63	-4	70	-9	80	-13	89	258	-20	84	-4	29	-9	34	-22	69
234	-2	72	-4	75	-9	84	-13	94	259	-11	79	5	31	1	32	-16	75
235	-1	74	-8	82	-10	87	-11	92	260	-6	73	7	42	11	31	-2	58
236	-9	91	-17	102	-34	127	-29	127	261	-4	74	1	59	5	52	1	58
237	-5	67	-9	77	-21	101	-21	100	262	-13	86	-2	58	-2	55	-13	78
238	-5	27	3	24	-3	39	-14	57	263	-13	96	-3	72	-2	67	-9	83
239	-3	39	0	42	-5	53	-12	67	264	-11	97	-3	81	0	73	-4	81
240	-2	44	0	48	-7	61	-11	71	265	-19	117	-11	97	-8	89	-13	99
241	0	46	0	55	-11	75	-10	77	266	-8	78	0	58	3	52	-13	81
242	-16	64	-6	59	-15	77	-21	85	267	-5	33	8	5	7	4	-6	38
243	-8	68	-7	71	-17	91	-16	88	268	-4	48	10	14	10	8	-13	62
244	-10	78	-12	83	-25	107	-23	104	269	-2	53	14	18	17	8	-6	58
245	-6	77	-11	87	-25	111	-15	98	270	-3	62	13	27	16	19	-2	59
246	-10	86	-30	119	-28	122	-37	127	271	-13	71	2	40	3	37	-16	73
247	-12	92	1	58	-11	76	-10	82	272	-10	79	0	58	-1	61	-10	76
248	-15	55	-3	29	-10	44	-8	48	273	-12	86	-3	70	-5	73	-13	86
249	-10	60	-1	36	-10	52	-8	61	274	-13	90	-6	79	-1	70	-9	83
250	-6	62	1	38	-10	57	-8	66	275	-14	97	-8	85	-4	78	-10	87
251	-4	65	2	43	-9	58	-7	70									

Table 9-22 – Values of variables m and n for ctxIdx from 277 to 337

ctxIdx	I and SI slices		Value of cabac_init_idc						ctxIdx	I and SI slices		Value of cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
277	-6	93	-13	106	-21	126	-22	127	308	-16	96	-1	51	-16	77	-10	67
278	-6	84	-16	106	-23	124	-25	127	309	-7	88	7	49	-2	64	1	68
279	-8	79	-10	87	-20	110	-25	120	310	-8	85	8	52	2	61	0	77
280	0	66	-21	114	-26	126	-27	127	311	-7	85	9	41	-6	67	2	64
281	-1	71	-18	110	-25	124	-19	114	312	-9	85	6	47	-3	64	0	68
282	0	62	-14	98	-17	105	-23	117	313	-13	88	2	55	2	57	-5	78
283	-2	60	-22	110	-27	121	-25	118	314	4	66	13	41	-3	65	7	55
284	-2	59	-21	106	-27	117	-26	117	315	-3	77	10	44	-3	66	5	59
285	-5	75	-18	103	-17	102	-24	113	316	-3	76	6	50	0	62	2	65
286	-3	62	-21	107	-26	117	-28	118	317	-6	76	5	53	9	51	14	54
287	-4	58	-23	108	-27	116	-31	120	318	10	58	13	49	-1	66	15	44
288	-9	66	-26	112	-33	122	-37	124	319	-1	76	4	63	-2	71	5	60
289	-1	79	-10	96	-10	95	-10	94	320	-1	83	6	64	-2	75	2	70
290	0	71	-12	95	-14	100	-15	102	321	-7	99	-2	69	-1	70	-2	76
291	3	68	-5	91	-8	95	-10	99	322	-14	95	-2	59	-9	72	-18	86
292	10	44	-9	93	-17	111	-13	106	323	2	95	6	70	14	60	12	70
293	-7	62	-22	94	-28	114	-50	127	324	0	76	10	44	16	37	5	64
294	15	36	-5	86	-6	89	-5	92	325	-5	74	9	31	0	47	-12	70
295	14	40	9	67	-2	80	17	57	326	0	70	12	43	18	35	11	55
296	16	27	-4	80	-4	82	-5	86	327	-11	75	3	53	11	37	5	56
297	12	29	-10	85	-9	85	-13	94	328	1	68	14	34	12	41	0	69
298	1	44	-1	70	-8	81	-12	91	329	0	65	10	38	10	41	2	65
299	20	36	7	60	-1	72	-2	77	330	-14	73	-3	52	2	48	-6	74
300	18	32	9	58	5	64	0	71	331	3	62	13	40	12	41	5	54
301	5	42	5	61	1	67	-1	73	332	4	62	17	32	13	41	7	54
302	1	48	12	50	9	56	4	64	333	-1	68	7	44	0	59	-6	76
303	10	62	15	50	0	69	-7	81	334	-13	75	7	38	3	50	-11	82
304	17	46	18	49	1	69	5	64	335	11	55	13	50	19	40	-2	77
305	9	64	17	54	7	69	15	57	336	5	64	10	57	3	66	-2	77
306	-12	104	10	41	-7	69	1	67	337	12	70	26	43	18	50	25	42
307	-11	97	7	46	-6	67	0	68									

Table 9-23 – Values of variables m and n for ctxIdx from 338 to 398

ctxIdx	I and SI slices		Value of cabac_init_idc						ctxIdx	I and SI slices		Value of cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
338	15	6	14	11	19	-6	17	-13	369	32	-26	31	-4	40	-37	37	-17
339	6	19	11	14	18	-6	16	-9	370	37	-30	27	6	38	-30	32	1
340	7	16	9	11	14	0	17	-12	371	44	-32	34	8	46	-33	34	15
341	12	14	18	11	26	-12	27	-21	372	34	-18	30	10	42	-30	29	15
342	18	13	21	9	31	-16	37	-30	373	34	-15	24	22	40	-24	24	25
343	13	11	23	-2	33	-25	41	-40	374	40	-15	33	19	49	-29	34	22
344	13	15	32	-15	33	-22	42	-41	375	33	-7	22	32	38	-12	31	16
345	15	16	32	-15	37	-28	48	-47	376	35	-5	26	31	40	-10	35	18
346	12	23	34	-21	39	-30	39	-32	377	33	0	21	41	38	-3	31	28
347	13	23	39	-23	42	-30	46	-40	378	38	2	26	44	46	-5	33	41
348	15	20	42	-33	47	-42	52	-51	379	33	13	23	47	31	20	36	28
349	14	26	41	-31	45	-36	46	-41	380	23	35	16	65	29	30	27	47
350	14	44	46	-28	49	-34	52	-39	381	13	58	14	71	25	44	21	62
351	17	40	38	-12	41	-17	43	-19	382	29	-3	8	60	12	48	18	31
352	17	47	21	29	32	9	32	11	383	26	0	6	63	11	49	19	26
353	24	17	45	-24	69	-71	61	-55	384	22	30	17	65	26	45	36	24
354	21	21	53	-45	63	-63	56	-46	385	31	-7	21	24	22	22	24	23
355	25	22	48	-26	66	-64	62	-50	386	35	-15	23	20	23	22	27	16
356	31	27	65	-43	77	-74	81	-67	387	34	-3	26	23	27	21	24	30
357	22	29	43	-19	54	-39	45	-20	388	34	3	27	32	33	20	31	29
358	19	35	39	-10	52	-35	35	-2	389	36	-1	28	23	26	28	22	41
359	14	50	30	9	41	-10	28	15	390	34	5	28	24	30	24	22	42
360	10	57	18	26	36	0	34	1	391	32	11	23	40	27	34	16	60
361	7	63	20	27	40	-1	39	1	392	35	5	24	32	18	42	15	52
362	-2	77	0	57	30	14	30	17	393	34	12	28	29	25	39	14	60
363	-4	82	-14	82	28	26	20	38	394	39	11	23	42	18	50	3	78
364	-3	94	-5	75	23	37	18	45	395	30	29	19	57	12	70	-16	123
365	9	69	-19	97	12	55	15	54	396	34	26	22	53	21	54	21	53
366	-12	109	-35	125	11	65	0	79	397	29	39	22	61	14	71	22	56
367	36	-35	27	0	37	-33	36	-16	398	19	66	11	86	11	83	25	61
368	36	-34	28	0	39	-36	37	-14									

Table 9-24 – Values of variables m and n for ctxIdx from 402 to 459

ctxIdx	I slices		Value of cabac_init_idc						ctxIdx	I slices		Value of cabac_init_idc					
			0		1		2					0		1		2	
	m	n	m	n	m	n	m	n		m	n	m	n	m	n	m	n
402	-17	120	-4	79	-5	85	-3	78	431	-2	55	-12	56	-9	57	-12	59
403	-20	112	-7	71	-6	81	-8	74	432	0	61	-6	60	-6	63	-8	63
404	-18	114	-5	69	-10	77	-9	72	433	1	64	-5	62	-4	65	-9	67
405	-11	85	-9	70	-7	81	-10	72	434	0	68	-8	66	-4	67	-6	68
406	-15	92	-8	66	-17	80	-18	75	435	-9	92	-8	76	-7	82	-10	79
407	-14	89	-10	68	-18	73	-12	71	436	-14	106	-5	85	-3	81	-3	78
408	-26	71	-19	73	-4	74	-11	63	437	-13	97	-6	81	-3	76	-8	74
409	-15	81	-12	69	-10	83	-5	70	438	-15	90	-10	77	-7	72	-9	72
410	-14	80	-16	70	-9	71	-17	75	439	-12	90	-7	81	-6	78	-10	72
411	0	68	-15	67	-9	67	-14	72	440	-18	88	-17	80	-12	72	-18	75
412	-14	70	-20	62	-1	61	-16	67	441	-10	73	-18	73	-14	68	-12	71
413	-24	56	-19	70	-8	66	-8	53	442	-9	79	-4	74	-3	70	-11	63
414	-23	68	-16	66	-14	66	-14	59	443	-14	86	-10	83	-6	76	-5	70
415	-24	50	-22	65	0	59	-9	52	444	-10	73	-9	71	-5	66	-17	75
416	-11	74	-20	63	2	59	-11	68	445	-10	70	-9	67	-5	62	-14	72
417	23	-13	9	-2	17	-10	9	-2	446	-10	69	-1	61	0	57	-16	67
418	26	-13	26	-9	32	-13	30	-10	447	-5	66	-8	66	-4	61	-8	53
419	40	-15	33	-9	42	-9	31	-4	448	-9	64	-14	66	-9	60	-14	59
420	49	-14	39	-7	49	-5	33	-1	449	-5	58	0	59	1	54	-9	52
421	44	3	41	-2	53	0	33	7	450	2	59	2	59	2	58	-11	68
422	45	6	45	3	64	3	31	12	451	21	-10	21	-13	17	-10	9	-2
423	44	34	49	9	68	10	37	23	452	24	-11	33	-14	32	-13	30	-10
424	33	54	45	27	66	27	31	38	453	28	-8	39	-7	42	-9	31	-4
425	19	82	36	59	47	57	20	64	454	28	-1	46	-2	49	-5	33	-1
426	-3	75	-6	66	-5	71	-9	71	455	29	3	51	2	53	0	33	7
427	-1	23	-7	35	0	24	-7	37	456	29	9	60	6	64	3	31	12
428	1	34	-7	42	-1	36	-8	44	457	35	20	61	17	68	10	37	23
429	1	43	-8	45	-2	42	-11	49	458	29	36	55	34	66	27	31	38
430	0	54	-5	48	-2	52	-10	56	459	14	67	42	62	47	57	20	64

9.3.1.2 Initialisation process for the arithmetic decoding engine

This process is invoked before decoding the first macroblock of a slice or after the decoding of any pcm_alignment_zero_bit and all pcm_sample_luma and pcm_sample_chroma data for a macroblock of type I_PCM.

Outputs of this process are the initialised decoding engine registers codIRange and codIOffset both in 16 bit register precision.

The status of the arithmetic decoding engine is represented by the variables codIRange and codIOffset. In the initialisation procedure of the arithmetic decoding process, codIRange is set equal to 0x01FE and codIOffset is set equal

to the value returned from `read_bits(9)` interpreted as a 9 bit binary representation of an unsigned integer with most significant bit written first.

The bitstream shall not contain data that results in a value of `codIOffset` being equal to 0x01FE or 0x01FF.

NOTE – The description of the arithmetic decoding engine in this Recommendation | International Standard utilizes 16 bit register precision. However, the minimum register precision for the variables `codIRange` and `codIOffset` is 9 bits.

9.3.2 Binarization process

Input to this process is a request for a syntax element.

Output of this process is the binarization of the syntax element, `maxBinIdxCtx`, `ctxIdxOffset`, and `bypassFlag`.

Table 9-25 specifies the type of binarization process, `maxBinIdxCtx`, and `ctxIdxOffset` associated with each syntax element.

The specification of the unary (U) binarization process, the truncated unary (TU) binarization process, the concatenated unary / k-th order Exp-Golomb (UEGk) binarization process, and the fixed-length (FL) binarization process are given in subclauses 9.3.2.1 to 9.3.2.4, respectively. Other binarizations are specified in subclauses 9.3.2.5 to 9.3.2.7.

Except for I slices, the binarizations for the syntax element `mb_type` as specified in subclause 9.3.2.5 consist of bin strings given by a concatenation of prefix and suffix bit strings. The UEGk binarization as specified in 9.3.2.3, which is used for the binarization of the syntax elements `mvd_IX` ($X = 0, 1$) and `coeff_abs_level_minus1`, and the binarization of the `coded_block_pattern` also consist of a concatenation of prefix and suffix bit strings. For these binarization processes, the prefix and the suffix bit string are separately indexed using the `binIdx` variable as specified further in subclause 9.3.3. The two sets of prefix bit strings and suffix bit strings are referred to as the binarization prefix part and the binarization suffix part, respectively.

Associated with each binarization or binarization part of a syntax element is a specific value of the context index offset (`ctxIdxOffset`) variable and a specific value of the `maxBinIdxCtx` variable as given in Table 9-25. When two values for each of these variables are specified for one syntax element in Table 9-25, the value in the upper row is related to the prefix part while the value in the lower row is related to the suffix part of the binarization of the corresponding syntax element.

The use of the `DecodeBypass` process and the variable `bypassFlag` is derived as follows.

- If no value is assigned to `ctxIdxOffset` for the corresponding binarization or binarization part in Table 9-25 labelled as “na”, all bins of the bit strings of the corresponding binarization or of the binarization prefix/suffix part are decoded by invoking the `DecodeBypass` process as specified in subclause 9.3.3.2.3. In such a case, `bypassFlag` is set equal to 1, where `bypassFlag` is used to indicate that for parsing the value of the bin from the bitstream the `DecodeBypass` process is applied.
- Otherwise, for each possible value of `binIdx` up to the specified value of `maxBinIdxCtx` given in Table 9-25, a specific value of the variable `ctxIdx` is further specified in subclause 9.3.3. `bypassFlag` is set equal to 0.

The possible values of the context index `ctxIdx` are in the range 0 to 459, inclusive. The value assigned to `ctxIdxOffset` specifies the lower value of the range of `ctxIdx` assigned to the corresponding binarization or binarization part of a syntax element.

`ctxIdx = ctxIdxOffset = 276` is assigned to the syntax element `end_of_slice_flag` and the bin of `mb_type`, which specifies the I_PCM macroblock type as further specified in subclause 9.3.3.1. For parsing the value of the corresponding bin from the bitstream, the arithmetic decoding process for decisions before termination (`DecodeTerminate`) as specified in subclause 9.3.3.2.4 is applied.

NOTE – The bins of `mb_type` in I slices and the bins of the suffix for `mb_type` in SI slices that correspond to the same value of `binIdx` share the same `ctxIdx`. The last bin of the prefix of `mb_type` and the first bin of the suffix of `mb_type` in P, SP, and B slices may share the same `ctxIdx`.

Table 9-25 – Syntax elements and associated types of binarization, maxBinIdxCtx, and ctxIdxOffset

Syntax element	Type of binarization	maxBinIdxCtx	ctxIdxOffset
mb_type (SI slices only)	prefix and suffix as specified in subclause 9.3.2.5	prefix: 0 suffix: 6	prefix: 0 suffix: 3
mb_type (I slices only)	as specified in subclause 9.3.2.5	6	3
mb_skip_flag (P, SP slices only)	FL, cMax=1	0	11
mb_type (P, SP slices only)	prefix and suffix as specified in subclause 9.3.2.5	prefix: 2 suffix: 5	prefix: 14 suffix: 17
sub_mb_type (P, SP slices only)	as specified in subclause 9.3.2.5	2	21
mb_skip_flag (B slices only)	FL, cMax=1	0	24
mb_type (B slices only)	prefix and suffix as specified in subclause 9.3.2.5	prefix: 3 suffix: 5	prefix: 27 suffix: 32
sub_mb_type (B slices only)	as specified in subclause 9.3.2.5	3	36
mvd_10[][0], mvd_11[][0]	prefix and suffix as given by UEG3 with signedValFlag=1, uCoff=9	prefix: 4 suffix: na	prefix: 40 suffix: na (uses DecodeBypass)
mvd_10[][1], mvd_11[][1]		prefix: 4 suffix: na	prefix: 47 suffix: na (uses DecodeBypass)
ref_idx_10, ref_idx_11	U	2	54
mb_qp_delta	as specified in subclause 9.3.2.7	2	60
intra_chroma_pred_mode	TU, cMax=3	1	64
prev_intra4x4_pred_mode_flag, prev_intra8x8_pred_mode_flag	FL, cMax=1	0	68
rem_intra4x4_pred_mode, rem_intra8x8_pred_mode	FL, cMax=7	0	69
mb_field_decoding_flag	FL, cMax=1	0	70
coded_block_pattern	prefix and suffix as specified in subclause 9.3.2.6	prefix: 3 suffix: 1	prefix: 73 suffix: 77
coded_block_flag	FL, cMax=1	0	85
significant_coeff_flag (frame coded blocks with ctxBlockCat < 5)	FL, cMax=1	0	105
last_significant_coeff_flag (frame coded blocks with ctxBlockCat < 5)	FL, cMax=1	0	166
coeff_abs_level_minus1 (blocks with ctxBlockCat < 5)	prefix and suffix as given by UEG0 with signedValFlag=0, uCoff=14	prefix: 1 suffix: na	prefix: 227 suffix: na, (uses DecodeBypass)
coeff_sign_flag	FL, cMax=1	0	na, (uses DecodeBypass)
end_of_slice_flag	FL, cMax=1	0	276
significant_coeff_flag (field coded blocks with ctxBlockCat < 5)	FL, cMax=1	0	277

Syntax element	Type of binarization	maxBinIdxCtx	ctxIdxOffset
last_significant_coeff_flag (field coded blocks with ctxBlockCat < 5)	FL, cMax=1	0	338
transform_size_8x8_flag	FL, cMax=1	0	399
significant_coeff_flag (frame coded blocks with ctxBlockCat == 5)	FL, cMax=1	0	402
last_significant_coeff_flag (frame coded blocks with ctxBlockCat == 5)	FL, cMax=1	0	417
coeff_abs_level_minus1 (blocks with ctxBlockCat == 5)	prefix and suffix as given by UEG0 with signedValFlag=0, uCoff=14	prefix: 1 suffix: na	prefix: 426 suffix: na, (uses DecodeBypass)
significant_coeff_flag (field coded blocks with ctxBlockCat == 5)	FL, cMax=1	0	436
last_significant_coeff_flag (field coded blocks with ctxBlockCat == 5)	FL, cMax=1	0	451

9.3.2.1 Unary (U) binarization process

Input to this process is a request for a U binarization for a syntax element.

Output of this process is the U binarization of the syntax element.

The bin string of a syntax element having (unsigned integer) value synElVal is a bit string of length synElVal + 1 indexed by BinIdx. The bins for binIdx less than synElVal are equal to 1. The bin with binIdx equal to synElVal is equal to 0.

Table 9-26 illustrates the bin strings of the unary binarization for a syntax element.

Table 9-26 – Bin string of the unary binarization (informative)

Value of syntax element	Bin string					
0 (I_NxN)	0					
1	1	0				
2	1	1	0			
3	1	1	1	0		
4	1	1	1	1	0	
5	1	1	1	1	1	0
...						
binIdx	0	1	2	3	4	5

9.3.2.2 Truncated unary (TU) binarization process

Input to this process is a request for a TU binarization for a syntax element and cMax.

Output of this process is the TU binarization of the syntax element.

For syntax element (unsigned integer) values less than cMax, the U binarization process as specified in subclause 9.3.2.1 is invoked. For the syntax element value equal to cMax the bin string is a bit string of length cMax with all bins being equal to 1.

NOTE – TU binarization is always invoked with a cMax value equal to the largest possible value of the syntax element being decoded.

9.3.2.3 Concatenated unary/ k-th order Exp-Golomb (UEGk) binarization process

Input to this process is a request for a UEGk binarization for a syntax element, signedValFlag and uCoff.

Output of this process is the UEGk binarization of the syntax element.

A UEGk bin string is a concatenation of a prefix bit string and a suffix bit string. The prefix of the binarization is specified by invoking the TU binarization process for the prefix part $\text{Min}(\text{uCoff}, \text{Abs}(\text{synElVal}))$ of a syntax element value synElVal as specified in subclause 9.3.2.2 with cMax = uCoff, where uCoff > 0.

The UEGk bin string is derived as follows.

- If one of the following is true, the bin string of a syntax element having value synElVal consists only of a prefix bit string,
 - signedValFlag is equal to 0 and the prefix bit string is not equal to the bit string of length uCoff with all bits equal to 1.
 - signedValFlag is equal to 1 and the prefix bit string is equal to the bit string that consists of a single bit with value equal to 0.
- Otherwise, the bin string of the UEGk suffix part of a syntax element value synElVal is specified by a process equivalent to the following pseudo-code:

```
if( Abs( synElVal ) >= uCoff ) {
    sufS = Abs( synElVal ) - uCoff
    stopLoop = 0
    do {
        if( sufS >= ( 1 << k ) ) {
            put( 1 )
            sufS = sufS - ( 1 << k )
            k++
        } else {
            put( 0 )
            while( k-- )
                put( ( sufS >> k ) & 0x01 )
            stopLoop = 1
        }
    } while( !stopLoop )
}
if( signedValFlag && synElVal != 0 )
    if( synElVal > 0 )
        put( 0 )
    else
        put( 1 )
```

NOTE – The specification for the k-th order Exp-Golomb (EGk) code uses 1's and 0's in reverse meaning for the unary part of the Exp-Golomb code of 0-th order as specified in subclause 9.1.

9.3.2.4 Fixed-length (FL) binarization process

Input to this process is a request for a FL binarization for a syntax element and cMax.

Output of this process is the FL binarization of the syntax element.

FL binarization is constructed by using an fixedLength-bit unsigned integer bin string of the syntax element value, where $\text{fixedLength} = \text{Ceil}(\text{Log}_2(\text{cMax} + 1))$. The indexing of bins for the FL binarization is such that the binIdx = 0 relates to the least significant bit with increasing values of binIdx towards the most significant bit.

9.3.2.5 Binarization process for macroblock type and sub-macroblock type

Input to this process is a request for a binarization for syntax elements mb_type or sub_mb_type.

Output of this process is the binarization of the syntax element.

The binarization scheme for decoding of macroblock type in I slices is specified in Table 9-27.

For macroblock types in SI slices, the binarization consists of bin strings specified as a concatenation of a prefix and a suffix bit string as follows.

The prefix bit string consists of a single bit, which is specified by $b_0 = ((mb_type == SI) ? 0 : 1)$. For the syntax element value for which b_0 is equal to 0, the bin string only consists of the prefix bit string. For the syntax element value for which b_0 is equal to 1, the binarization is given by concatenating the prefix b_0 and the suffix bit string as specified in Table 9-27 for macroblock type in I slices indexed by subtracting 1 from the value of mb_type in SI slices.

Table 9-27 – Binarization for macroblock types in I slices

Value (name) of mb_type	Bin string						
0 (I_4x4)	0						
1 (I_16x16_0_0_0)	1	0	0	0	0	0	
2 (I_16x16_1_0_0)	1	0	0	0	0	1	
3 (I_16x16_2_0_0)	1	0	0	0	1	0	
4 (I_16x16_3_0_0)	1	0	0	0	1	1	
5 (I_16x16_0_1_0)	1	0	0	1	0	0	0
6 (I_16x16_1_1_0)	1	0	0	1	0	0	1
7 (I_16x16_2_1_0)	1	0	0	1	0	1	0
8 (I_16x16_3_1_0)	1	0	0	1	0	1	1
9 (I_16x16_0_2_0)	1	0	0	1	1	0	0
10 (I_16x16_1_2_0)	1	0	0	1	1	0	1
11 (I_16x16_2_2_0)	1	0	0	1	1	1	0
12 (I_16x16_3_2_0)	1	0	0	1	1	1	1
13 (I_16x16_0_0_1)	1	0	1	0	0	0	
14 (I_16x16_1_0_1)	1	0	1	0	0	1	
15 (I_16x16_2_0_1)	1	0	1	0	1	0	
16 (I_16x16_3_0_1)	1	0	1	0	1	1	
17 (I_16x16_0_1_1)	1	0	1	1	0	0	0
18 (I_16x16_1_1_1)	1	0	1	1	0	0	1
19 (I_16x16_2_1_1)	1	0	1	1	0	1	0
20 (I_16x16_3_1_1)	1	0	1	1	0	1	1
21 (I_16x16_0_2_1)	1	0	1	1	1	0	0
22 (I_16x16_1_2_1)	1	0	1	1	1	0	1
23 (I_16x16_2_2_1)	1	0	1	1	1	1	0
24 (I_16x16_3_2_1)	1	0	1	1	1	1	1
25 (I_PCM)	1	1					
binIdx	0	1	2	3	4	5	6

The binarization schemes for P macroblock types in P and SP slices and for B macroblocks in B slices are specified in Table 9-28.

The bin string for I macroblock types in P and SP slices corresponding to mb_type values 5 to 30 consists of a concatenation of a prefix, which consists of a single bit with value equal to 1 as specified in Table 9-28 and a suffix as specified in Table 9-27, indexed by subtracting 5 from the value of mb_type .

mb_type equal to 4 (P_8x8ref0) is not allowed.

For I macroblock types in B slices (mb_type values 23 to 48) the binarization consists of bin strings specified as a concatenation of a prefix bit string as specified in Table 9-28 and suffix bit strings as specified in Table 9-27, indexed by subtracting 23 from the value of mb_type.

Table 9-28 – Binarization for macroblock types in P, SP, and B slices

Slice type	Value (name) of mb_type	Bin string							
P, SP slice	0 (P_L0_16x16)	0	0	0					
	1 (P_L0_L0_16x8)	0	1	1					
	2 (P_L0_L0_8x16)	0	1	0					
	3 (P_8x8)	0	0	1					
	4 (P_8x8ref0)	na							
	5 to 30 (Intra, prefix only)	1							
B slice	0 (B_Direct_16x16)	0							
	1 (B_L0_16x16)	1	0	0					
	2 (B_L1_16x16)	1	0	1					
	3 (B_Bi_16x16)	1	1	0	0	0	0		
	4 (B_L0_L0_16x8)	1	1	0	0	0	1		
	5 (B_L0_L0_8x16)	1	1	0	0	1	0		
	6 (B_L1_L1_16x8)	1	1	0	0	1	1		
	7 (B_L1_L1_8x16)	1	1	0	1	0	0		
	8 (B_L0_L1_16x8)	1	1	0	1	0	1		
	9 (B_L0_L1_8x16)	1	1	0	1	1	0		
	10 (B_L1_L0_16x8)	1	1	0	1	1	1		
	11 (B_L1_L0_8x16)	1	1	1	1	1	0		
	12 (B_L0_Bi_16x8)	1	1	1	0	0	0	0	
	13 (B_L0_Bi_8x16)	1	1	1	0	0	0	1	
	14 (B_L1_Bi_16x8)	1	1	1	0	0	1	0	
	15 (B_L1_Bi_8x16)	1	1	1	0	0	1	1	
	16 (B_Bi_L0_16x8)	1	1	1	0	1	0	0	
	17 (B_Bi_L0_8x16)	1	1	1	0	1	0	1	
	18 (B_Bi_L1_16x8)	1	1	1	0	1	1	0	
	19 (B_Bi_L1_8x16)	1	1	1	0	1	1	1	
	20 (B_Bi_Bi_16x8)	1	1	1	1	0	0	0	
	21 (B_Bi_Bi_8x16)	1	1	1	1	0	0	1	
	22 (B_8x8)	1	1	1	1	1	1		
	23 to 48 (Intra, prefix only)	1	1	1	1	0	1		
binIdx		0	1	2	3	4	5	6	

For P, SP, and B slices the specification of the binarization for sub_mb_type is given in Table 9-29.

Table 9-29 – Binarization for sub-macroblock types in P, SP, and B slices

Slice type	Value (name) of sub_mb_type	Bin string					
P, SP slice	0 (P_L0_8x8)	1					
	1 (P_L0_8x4)	0	0				
	2 (P_L0_4x8)	0	1	1			
	3 (P_L0_4x4)	0	1	0			
B slice	0 (B_Direct_8x8)	0					
	1 (B_L0_8x8)	1	0	0			
	2 (B_L1_8x8)	1	0	1			
	3 (B_Bi_8x8)	1	1	0	0	0	
	4 (B_L0_8x4)	1	1	0	0	1	
	5 (B_L0_4x8)	1	1	0	1	0	
	6 (B_L1_8x4)	1	1	0	1	1	
	7 (B_L1_4x8)	1	1	1	0	0	0
	8 (B_Bi_8x4)	1	1	1	0	0	1
	9 (B_Bi_4x8)	1	1	1	0	1	0
	10 (B_L0_4x4)	1	1	1	0	1	1
	11 (B_L1_4x4)	1	1	1	1	0	
	12 (B_Bi_4x4)	1	1	1	1	1	
binIdx		0	1	2	3	4	5

9.3.2.6 Binarization process for coded block pattern

Input to this process is a request for a binarization for the syntax element coded_block_pattern.

Output of this process is the binarization of the syntax element.

The binarization of coded_block_pattern consists of a prefix part and (when present) a suffix part. The prefix part of the binarization is given by the FL binarization of CodedBlockPatternLuma with cMax = 15. When chroma_format_idc is not equal to 0 (monochrome), the suffix part is present and consists of the TU binarization of CodedBlockPatternChroma with cMax = 2. The relationship between the value of the syntax element coded_block_pattern and the values of CodedBlockPatternLuma and CodedBlockPatternChroma is given as specified in subclause 7.4.5.

9.3.2.7 Binarization process for mb_qp_delta

Input to this process is a request for a binarization for the syntax element mb_qp_delta.

Output of this process is the binarization of the syntax element.

The bin string of mb_qp_delta is derived by the U binarization of the mapped value of the syntax element mb_qp_delta, where the assignment rule between the signed value of mb_qp_delta and its mapped value is given as specified in Table 9-3.

9.3.3 Decoding process flow

Input to this process is a binarization of the requested syntax element, maxBinIdxCtx, bypassFlag and ctxIdxOffset as specified in subclause 9.3.2.

Output of this process is the value of the syntax element.

This process specifies how each bit of a bit string is parsed for each syntax element.

After parsing each bit, the resulting bit string is compared to all bin strings of the binarization of the syntax element and the following applies.

- If the bit string is equal to one of the bin strings, the corresponding value of the syntax element is the output.
- Otherwise (the bit string is not equal to one of the bin strings), the next bit is parsed.

While parsing each bin, the variable `binIdx` is incremented by 1 starting with `binIdx` being set equal to 0 for the first bin.

When the binarization of the corresponding syntax element consists of a prefix and a suffix binarization part,, the variable `binIdx` is set equal to 0 for the first bin of each part of the bin string (prefix part or suffix part). In this case, after parsing the prefix bit string, the parsing process of the suffix bit string related to the binarizations specified in subclauses 9.3.2.3 and 9.3.2.5 is invoked depending on the resulting prefix bit string as specified in subclauses 9.3.2.3 and 9.3.2.5. Note that for the binarization of the syntax element `coded_block_pattern`, the suffix bit string is present regardless of the prefix bit string of length 4 as specified in subclause 9.3.2.6.

Depending on the variable `bypassFlag`, the following applies.

- If `bypassFlag` is equal to 1, the bypass decoding process as specified in subclause 9.3.3.2.3 is applied for parsing the value of the bins from the bitstream.
- Otherwise (`bypassFlag` is equal to 0), the parsing of each bin is specified by the following two ordered steps:
 1. Given `binIdx`, `maxBinIdxCtx` and `ctxIdxOffset`, `ctxIdx` is derived as specified in subclause 9.3.3.1.
 2. Given `ctxIdx`, the value of the bin from the bitstream as specified in subclause 9.3.3.2 is decoded.

9.3.3.1 Derivation process for `ctxIdx`

Inputs to this process are `binIdx`, `maxBinIdxCtx` and `ctxIdxOffset`.

Output of this process is `ctxIdx`.

Table 9-30 shows the assignment of `ctxIdx` increments (`ctxIdxInc`) to `binIdx` for all `ctxIdxOffset` values except those related to the syntax elements `coded_block_flag`, `significant_coeff_flag`, `last_significant_coeff_flag`, and `coeff_abs_level_minus1`.

The `ctxIdx` to be used with a specific `binIdx` is specified by first determining the `ctxIdxOffset` associated with the given bin string or part thereof. The `ctxIdx` is determined as follows.

- If the `ctxIdxOffset` is listed in Table 9-30, the `ctxIdx` for a `binIdx` is the sum of `ctxIdxOffset` and `ctxIdxInc`, which is found in Table 9-30. When more than one value is listed in Table 9-30 for a `binIdx`, the assignment process for `ctxIdxInc` for that `binIdx` is further specified in the subclauses given in parenthesis of the corresponding table entry.
- Otherwise (`ctxIdxOffset` is not listed in Table 9-30), the `ctxIdx` is specified to be the sum of the following terms: `ctxIdxOffset` and `ctxIdxBlockCatOffset(ctxBlockCat)` as specified in Table 9-31 and `ctxIdxInc(ctxBlockCat)`. Subclause 9.3.3.1.3 specifies which `ctxBlockCat` is used. Subclause 9.3.3.1.9 specifies the assignment of `ctxIdxInc(ctxBlockCat)` for `coded_block_flag`, and subclause 9.3.3.1.3 specifies the assignment of `ctxIdxInc(ctxBlockCat)` for `significant_coeff_flag`, `last_significant_coeff_flag`, and `coeff_abs_level_minus1`.

All bins with `binIdx` greater than `maxBinIdxCtx` are parsed using the value of `ctxIdx` being assigned to `binIdx` equal to `maxBinIdxCtx`.

All entries in Table 9-30 labelled with “na” correspond to values of `binIdx` that do not occur for the corresponding `ctxIdxOffset`.

`ctxIdx = 276` is assigned to the `binIdx` of `mb_type` indicating the `I_PCM` mode. For parsing the value of the corresponding bins from the bitstream, the arithmetic decoding process for decisions before termination as specified in subclause 9.3.3.2.4 is applied.

Table 9-30 – Assignment of ctxIdxInc to binIdx for all ctxIdxOffset values except those related to the syntax elements coded_block_flag, significant_coeff_flag, last_significant_coeff_flag, and coeff_abs_level_minus1

ctxIdxOffset	binIdx						
	0	1	2	3	4	5	>= 6
0	0,1,2 (subclause 9.3.3.1.1.3)	na	na	na	na	na	na
3	0,1,2 (subclause 9.3.3.1.1.3)	ctxIdx=276	3	4	5,6 (subclause 9.3.3.1.2)	6,7 (subclause 9.3.3.1.2)	7
11	0,1,2 (subclause 9.3.3.1.1.1)	na	na	na	na	na	na
14	0	1	2,3 (subclause 9.3.3.1.2)	na	na	na	na
17	0	ctxIdx=276	1	2	2,3 (subclause 9.3.3.1.2)	3	3
21	0	1	2	na	na	na	na
24	0,1,2 (subclause 9.3.3.1.1.1)	na	na	na	na	na	na
27	0,1,2 (subclause 9.3.3.1.1.3)	3	4,5 (subclause 9.3.3.1.2)	5	5	5	5
32	0	ctxIdx=276	1	2	2,3 (subclause 9.3.3.1.2)	3	3
36	0	1	2,3 (subclause 9.3.3.1.2)	3	3	3	na
40	0,1,2 (subclause 9.3.3.1.1.7)	3	4	5	6	6	6
47	0,1,2 (subclause 9.3.3.1.1.7)	3	4	5	6	6	6
54	0,1,2,3 (subclause 9.3.3.1.1.6)	4	5	5	5	5	5
60	0,1 (subclause 9.3.3.1.1.5)	2	3	3	3	3	3
64	0,1,2 (subclause 9.3.3.1.1.8)	3	3	na	na	na	na
68	0	na	na	na	na	na	na
69	0	0	0	na	na	na	na
70	0,1,2 (subclause 9.3.3.1.1.2)	na	na	na	na	na	na
73	0,1,2,3 (subclause 9.3.3.1.1.4)	0,1,2,3 (subclause 9.3.3.1.1.4)	0,1,2,3 (subclause 9.3.3.1.1.4)	0,1,2,3 (subclause 9.3.3.1.1.4)	na	na	na
77	0,1,2,3 (subclause 9.3.3.1.1.4)	4,5,6,7 (subclause 9.3.3.1.1.4)	na	na	na	na	na
276	0	na	na	na	na	na	na
399	0,1,2 (subclause 9.3.3.1.1.10)	na	na	na	na	na	na

Table 9-31 shows the values of `ctxIdxBlockCatOffset` depending on `ctxBlockCat` for the syntax elements `coded_block_flag`, `significant_coeff_flag`, `last_significant_coeff_flag`, and `coeff_abs_level_minus1`. The specification of `ctxBlockCat` is given in Table 9-33.

Table 9-31 – Assignment of `ctxIdxBlockCatOffset` to `ctxBlockCat` for syntax elements `coded_block_flag`, `significant_coeff_flag`, `last_significant_coeff_flag`, and `coeff_abs_level_minus1`

Syntax element	ctxBlockCat (as specified in Table 9-33)					
	0	1	2	3	4	5
<code>coded_block_flag</code>	0	4	8	12	16	na
<code>significant_coeff_flag</code>	0	15	29	44	47	0
<code>last_significant_coeff_flag</code>	0	15	29	44	47	0
<code>coeff_abs_level_minus1</code>	0	10	20	30	39	0

9.3.3.1.1 Assignment process of `ctxIdxInc` using neighbouring syntax elements

Subclause 9.3.3.1.1.1 specifies the derivation process of `ctxIdxInc` for the syntax element `mb_skip_flag`.

Subclause 9.3.3.1.1.2 specifies the derivation process of `ctxIdxInc` for the syntax element `mb_field_decoding_flag`.

Subclause 9.3.3.1.1.3 specifies the derivation process of `ctxIdxInc` for the syntax element `mb_type`.

Subclause 9.3.3.1.1.4 specifies the derivation process of `ctxIdxInc` for the syntax element `coded_block_pattern`.

Subclause 9.3.3.1.1.5 specifies the derivation process of `ctxIdxInc` for the syntax element `mb_qp_delta`.

Subclause 9.3.3.1.1.6 specifies the derivation process of `ctxIdxInc` for the syntax elements `ref_idx_l0` and `ref_idx_l1`.

Subclause 9.3.3.1.1.7 specifies the derivation process of `ctxIdxInc` for the syntax elements `mvd_l0` and `mvd_l1`.

Subclause 9.3.3.1.1.8 specifies the derivation process of `ctxIdxInc` for the syntax element `intra_chroma_pred_mode`.

Subclause 9.3.3.1.1.9 specifies the derivation process of `ctxIdxInc` for the syntax element `coded_block_flag`.

Subclause 9.3.3.1.1.10 specifies the derivation process of `ctxIdxInc` for the syntax element `transform_size_8x8_flag`.

9.3.3.1.1.1 Derivation process of `ctxIdxInc` for the syntax element `mb_skip_flag`

Output of this process is `ctxIdxInc`.

When `MbaffFrameFlag` is equal to 1 and `mb_field_decoding_flag` has not been decoded (yet) for the current macroblock pair with top macroblock address $2 * (\text{CurrMbAddr} / 2)$, the inference rule for the syntax element `mb_field_decoding_flag` as specified in subclause 7.4.4 is applied.

The derivation process for neighbouring macroblocks specified in subclause 6.4.8.1 is invoked and the output is assigned to `mbAddrA` and `mbAddrB`.

Let the variable `condTermFlagN` (with `N` being either `A` or `B`) be derived as follows.

- If `mbAddrN` is not available or `mb_skip_flag` for the macroblock `mbAddrN` is equal to 1, `condTermFlagN` is set equal to 0.
- Otherwise (`mbAddrN` is available and `mb_skip_flag` for the macroblock `mbAddrN` is equal to 0), `condTermFlagN` is set equal to 1.

The variable `ctxIdxInc` is derived by

$$\text{ctxIdxInc} = \text{condTermFlagA} + \text{condTermFlagB} \quad (9-1)$$

9.3.3.1.1.2 Derivation process of `ctxIdxInc` for the syntax element `mb_field_decoding_flag`

Output of this process is `ctxIdxInc`.

The derivation process for neighbouring macroblock addresses and their availability in MBAFF frames as specified in subclause 6.4.7 is invoked and the output is assigned to `mbAddrA` and `mbAddrB`.

When both macroblocks mbAddrN and mbAddrN + 1 have mb_type equal to P_Skip or B_Skip, the inference rule for the syntax element mb_field_decoding_flag as specified in subclause 7.4.4 is applied for the macroblock mbAddrN.

Let the variable condTermFlagN (with N being either A or B) be derived as follows.

- If any of the following conditions is true, condTermFlagN is set equal to 0,
 - mbAddrN is not available
 - the macroblock mbAddrN is a frame macroblock.
- Otherwise, condTermFlagN is set equal to 1.

The variable ctxIdxInc is derived by

$$\text{ctxIdxInc} = \text{condTermFlagA} + \text{condTermFlagB} \quad (9-2)$$

9.3.3.1.1.3 Derivation process of ctxIdxInc for the syntax element mb_type

Input to this process is ctxIdxOffset.

Output of this process is ctxIdxInc.

The derivation process for neighbouring macroblocks specified in subclause 6.4.8.1 is invoked and the output is assigned to mbAddrA and mbAddrB.

Let the variable condTermFlagN (with N being either A or B) be derived as follows.

- If any of the following conditions is true, condTermFlagN is set equal to 0
 - mbAddrN is not available
 - ctxIdxOffset is equal to 0 and mb_type for the macroblock mbAddrN is equal to SI
 - ctxIdxOffset is equal to 3 and mb_type for the macroblock mbAddrN is equal to I_NxN
 - ctxIdxOffset is equal to 27 and mb_type for the macroblock mbAddrN is equal to P_Skip, B_Skip, or B_Direct_16x16
- Otherwise, condTermFlagN is set equal to 1.

The variable ctxIdxInc is derived as

$$\text{ctxIdxInc} = \text{condTermFlagA} + \text{condTermFlagB} \quad (9-3)$$

9.3.3.1.1.4 Derivation process of ctxIdxInc for the syntax element coded_block_pattern

Inputs to this process are ctxIdxOffset and binIdx.

Output of this process is ctxIdxInc.

Depending on the value of the variable ctxIdxOffset, the following applies.

- If ctxIdxOffset is equal to 73, the following applies
 - The derivation process for neighbouring 8x8 luma blocks specified in subclause 6.4.8.2 is invoked with luma8x8BlkIdx = binIdx as input and the output is assigned to mbAddrA, mbAddrB, luma8x8BlkIdxA, and luma8x8BlkIdxB.
 - Let the variable condTermFlagN (with N being either A or B) be derived as follows.
 - If any of the following conditions are true, condTermFlagN is set equal to 0
 - mbAddrN is not available
 - mb_type for the macroblock mbAddrN is equal to I_PCM
 - the macroblock mbAddrN is not the current macroblock CurrMbAddr and the macroblock mbAddrN does not have mb_type equal to P_Skip or B_Skip, and ((CodedBlockPatternLuma >> luma8x8BlkIdxN) & 1) is not equal to 0 for the value of CodedBlockPatternLuma for the macroblock mbAddrN

- the macroblock mbAddrN is the current macroblock CurrMbAddr and the prior decoded bin value b_k of coded_block_pattern with $k = \text{luma8x8BlkIdxN}$ is not equal to 0.
- Otherwise, condTermFlagN is set equal to 1.
- The variable ctxIdxInc is derived as

$$\text{ctxIdxInc} = \text{condTermFlagA} + 2 * \text{condTermFlagB} \quad (9-4)$$

- Otherwise (ctxIdxOffset is equal to 77), the following applies.
 - The derivation process for neighbouring macroblocks specified in subclause 6.4.8.1 is invoked and the output is assigned to mbAddrA and mbAddrB.
 - Let the variable condTermFlagN (with N being either A or B) be derived as follows.
 - If mbAddrN is available and mb_type for the macroblock mbAddrN is equal to I_PCM, condTermFlagN is set equal to 1
 - Otherwise, if any of the following conditions is true, condTermFlagN is set equal to 0
 - mbAddrN is not available or the macroblock mbAddrN has mb_type equal to P_Skip or B_Skip
 - binIdx is equal to 0 and CodedBlockPatternChroma for the macroblock mbAddrN is equal to 0
 - binIdx is equal to 1 and CodedBlockPatternChroma for the macroblock mbAddrN is not equal to 2
 - Otherwise, condTermFlagN is set equal to 1.
 - The variable ctxIdxInc is derived as

$$\text{ctxIdxInc} = \text{condTermFlagA} + 2 * \text{condTermFlagB} + ((\text{binIdx} == 1) ? 4 : 0) \quad (9-5)$$

NOTE – When a macroblock uses an Intra_16x16 prediction mode, the values of CodedBlockPatternLuma and CodedBlockPatternChroma for the macroblock are derived from mb_type as specified in Table 7-11.

9.3.3.1.1.5 Derivation process of ctxIdxInc for the syntax element mb_qp_delta

Output of this process is ctxIdxInc.

Let prevMbAddr be the macroblock address of the macroblock that precedes the current macroblock in decoding order. When the current macroblock is the first macroblock of a slice, prevMbAddr is marked as not available.

Let the variable ctxIdxInc be derived as follows.

- If any of the following conditions is true, ctxIdxInc is set equal to 0
 - prevMbAddr is not available or the macroblock prevMbAddr has mb_type equal to P_Skip or B_Skip
 - mb_type of the macroblock prevMbAddr is equal to I_PCM
 - The macroblock prevMbAddr is not coded in Intra_16x16 prediction mode and both CodedBlockPatternLuma and CodedBlockPatternChroma for the macroblock prevMbAddr are equal to 0
 - mb_qp_delta for the macroblock prevMbAddr is equal to 0
- Otherwise, ctxIdxInc is set equal to 1.

9.3.3.1.1.6 Derivation process of ctxIdxInc for the syntax elements ref_idx_l0 and ref_idx_l1

Input to this process is mbPartIdx.

Output of this process is ctxIdxInc.

The interpretation of ref_idx_lX and Pred_LX within this subclause is specified as follows.

- If this process is invoked for the derivation of ref_idx_l0, ref_idx_lX is interpreted as ref_idx_l0 and Pred_LX is interpreted as Pred_L0.
- Otherwise (this process is invoked for the derivation of ref_idx_l1), ref_idx_lX is interpreted as ref_idx_l1 and Pred_LX is interpreted as Pred_L1.

Let currSubMbType be set equal to sub_mb_type[mbPartIdx].

The derivation process for neighbouring partitions specified in subclause 6.4.8.5 is invoked with mbPartIdx, currSubMbType, and subMbPartIdx = 0 as input and the output is assigned to mbAddrA\mbPartIdxA and mbAddrB\mbPartIdxB.

With ref_idx_IX[mbPartIdxN] (with N being either A or B) specifying the syntax element for the macroblock mbAddrN, let the variable refIdxZeroFlagN be derived as follows.

- If MbaffFrameFlag is equal to 1, the current macroblock is a frame macroblock, and the macroblock mbAddrN is a field macroblock

$$\text{refIdxZeroFlagN} = ((\text{ref_idx_IX}[\text{mbPartIdxN}] > 1) ? 0 : 1) \quad (9-6)$$

- Otherwise,

$$\text{refIdxZeroFlagN} = ((\text{ref_idx_IX}[\text{mbPartIdxN}] > 0) ? 0 : 1) \quad (9-7)$$

Let the variable predModeEqualFlagN be specified as follows.

- If the macroblock mbAddrN has mb_type equal to P_8x8 or B_8x8, the following applies.
 - If SubMbPredMode(sub_mb_type[mbPartIdxN]) is not equal to Pred_LX and not equal to BiPred, predModeEqualFlagN is set equal to 0, where sub_mb_type specifies the syntax element for the macroblock mbAddrN.
 - Otherwise, predModeEqualFlagN is set equal to 1.
- Otherwise, the following applies.
 - If MbPartPredMode(mb_type, mbPartIdxN) is not equal to Pred_LX and not equal to BiPred, predModeEqualFlagN is set equal to 0, where mb_type specifies the syntax element for the macroblock mbAddrN.
 - Otherwise, predModeEqualFlagN is set equal to 1.

Let the variable condTermFlagN (with N being either A or B) be derived as follows.

- If any of the following conditions is true, condTermFlagN is set equal to 0
 - mbAddrN is not available
 - the macroblock mbAddrN has mb_type equal to P_Skip or B_Skip
 - The macroblock mbAddrN is coded in Intra prediction mode
 - predModeEqualFlagN is equal to 0
 - refIdxZeroFlagN is equal to 1
- Otherwise, condTermFlagN is set equal to 1.

The variable ctxIdxInc is derived as

$$\text{ctxIdxInc} = \text{condTermFlagA} + 2 * \text{condTermFlagB} \quad (9-8)$$

9.3.3.1.1.7 Derivation process of ctxIdxInc for the syntax elements mvd_l0 and mvd_l1

Inputs to this process are mbPartIdx, subMbPartIdx, and ctxIdxOffset.

Output of this process is ctxIdxInc.

The interpretation of mvd_IX and Pred_LX within this subclause is specified as follows.

- If this process is invoked for the derivation of mvd_l0, mvd_IX is interpreted as mvd_l0 and Pred_LX is interpreted as Pred_L0.
- Otherwise (this process is invoked for the derivation of mvd_l1), mvd_IX is interpreted as mvd_l1 and Pred_LX is interpreted as Pred_L1.

Let currSubMbType be set equal to sub_mb_type[mbPartIdx].

The derivation process for neighbouring partitions specified in subclause 6.4.8.5 is invoked with mbPartIdx, currSubMbType, and subMbPartIdx as input and the output is assigned to mbAddrA\mbPartIdxA\subMbPartIdxA and mbAddrB\mbPartIdxB\subMbPartIdxB.

Let the variable compIdx be derived as follows.

- If ctxIdxOffset is equal to 40, compIdx is set equal to 0.
- Otherwise (ctxIdxOffset is equal to 47), compIdx is set equal to 1.

Let the variable predModeEqualFlagN be specified as follows.

- If the macroblock mbAddrN has mb_type equal to P_8x8 or B_8x8, the following applies.
 - If SubMbPredMode(sub_mb_type[mbPartIdxN]) is not equal to Pred_LX and not equal to BiPred, predModeEqualFlagN is set equal to 0, where sub_mb_type specifies the syntax element for the macroblock mbAddrN.
 - Otherwise, predModeEqualFlagN is set equal to 1.
- Otherwise, the following applies.
 - If MbPartPredMode(mb_type, mbPartIdxN) is not equal to Pred_LX and not equal to BiPred, predModeEqualFlagN is set equal to 0, where mb_type specifies the syntax element for the macroblock mbAddrN.
 - Otherwise, predModeEqualFlagN is set equal to 1.

Let the variable absMvdCompN (with N being either A or B) be derived as follows.

- If any of the following conditions is true, absMvdCompN is set equal to 0
 - mbAddrN is not available
 - the macroblock mbAddrN has mb_type equal to P_Skip or B_Skip
 - The macroblock mbAddrN is coded in an Intra prediction mode
 - predModeEqualFlagN is equal to 0
- Otherwise, the following applies
 - If compIdx is equal to 1, MbaffFrameFlag is equal to 1, the current macroblock is a frame macroblock, and the macroblock mbAddrN is a field macroblock

$$\text{absMvdCompN} = \text{Abs}(\text{mvd_IX}[\text{mbPartIdxN}][\text{subMbPartIdxN}][\text{compIdx}]) * 2 \quad (9-9)$$

- Otherwise, if compIdx is equal to 1, MbaffFrameFlag is equal to 1, the current macroblock is a field macroblock, and the macroblock mbAddrN is a frame macroblock

$$\text{absMvdCompN} = \text{Abs}(\text{mvd_IX}[\text{mbPartIdxN}][\text{subMbPartIdxN}][\text{compIdx}]) / 2 \quad (9-10)$$

- Otherwise,

$$\text{absMvdCompN} = \text{Abs}(\text{mvd_IX}[\text{mbPartIdxN}][\text{subMbPartIdxN}][\text{compIdx}]) \quad (9-11)$$

The variable ctxIdxInc is derived as follows

- If (absMvdCompA + absMvdCompB) is less than 3, ctxIdxInc is set equal to 0.
- Otherwise, if (absMvdCompA + absMvdCompB) is greater than 32, ctxIdxInc is set equal to 2.
- Otherwise ((absMvdCompA + absMvdCompB) is in the range of 3 to 32, inclusive), ctxIdxInc is set equal to 1.

9.3.3.1.1.8 Derivation process of ctxIdxInc for the syntax element intra_chroma_pred_mode

Output of this process is ctxIdxInc.

The derivation process for neighbouring macroblocks specified in subclause 6.4.8.1 is invoked and the output is assigned to mbAddrA and mbAddrB.

Let the variable `condTermFlagN` (with `N` being replaced by either `A` or `B`) be derived as follows.

- If any of the following conditions is true, `condTermFlagN` is set equal to 0
 - `mbAddrN` is not available
 - The macroblock `mbAddrN` is coded in Inter prediction mode
 - `mb_type` for the macroblock `mbAddrN` is equal to `I_PCM`
 - `intra_chroma_pred_mode` for the macroblock `mbAddrN` is equal to 0
- Otherwise, `condTermFlagN` is set equal to 1.

The variable `ctxIdxInc` is derived by

$$\text{ctxIdxInc} = \text{condTermFlagA} + \text{condTermFlagB} \quad (9-12)$$

9.3.3.1.1.9 Derivation process of `ctxIdxInc` for the syntax element `coded_block_flag`

Input to this process is `ctxBlockCat` and additional input is specified as follows.

- If `ctxBlockCat` is equal to 0, no additional input
- Otherwise, if `ctxBlockCat` is equal to 1 or 2, `luma4x4BlkIdx`
- Otherwise, if `ctxBlockCat` is equal to 3, the chroma component index `iCbCr`
- Otherwise (`ctxBlockCat` is equal to 4), `chroma4x4BlkIdx` and the chroma component index `iCbCr`

Output of this process is `ctxIdxInc(ctxBlockCat)`.

Let the variable `transBlockN` (with `N` being either `A` or `B`) be derived as follows.

- If `ctxBlockCat` is equal to 0, the following applies.
 - The derivation process for neighbouring macroblocks specified in subclause 6.4.8.1 is invoked and the output is assigned to `mbAddrN` (with `N` being either `A` or `B`).
 - The variable `transBlockN` is derived as follows.
 - If `mbAddrN` is available and the macroblock `mbAddrN` is coded in `Intra_16x16` prediction mode, the luma DC block of macroblock `mbAddrN` is assigned to `transBlockN`
 - Otherwise, `transBlockN` is marked as not available.
- Otherwise, if `ctxBlockCat` is equal to 1 or 2, the following applies.
 - The derivation process for neighbouring 4x4 luma blocks specified in subclause 6.4.8.3 is invoked with `luma4x4BlkIdx` as input and the output is assigned to `mbAddrN`, `luma4x4BlkIdxN` (with `N` being either `A` or `B`).
 - The variable `transBlockN` is derived as follows.
 - If `mbAddrN` is available, the macroblock `mbAddrN` does not have `mb_type` equal to `P_Skip`, `B_Skip`, or `I_PCM`, $((\text{CodedBlockPatternLuma} \gg (\text{luma4x4BlkIdxN} \gg 2)) \& 1)$ is not equal to 0 for the macroblock `mbAddrN`, and `transform_size_8x8_flag` is equal to 0 for the macroblock `mbAddrN`, the 4x4 luma block with index `luma4x4BlkIdxN` of macroblock `mbAddrN` is assigned to `transBlockN`.
 - Otherwise, if `mbAddrN` is available, the macroblock `mbAddrN` does not have `mb_type` equal to `P_Skip` or `B_Skip`, $((\text{CodedBlockPatternLuma} \gg (\text{luma4x4BlkIdxN} \gg 2)) \& 1)$ is not equal to 0 for the macroblock `mbAddrN`, and `transform_size_8x8_flag` is equal to 1 for the macroblock `mbAddrN`, the 8x8 luma block with index $(\text{luma4x4BlkIdxN} \gg 2)$ of macroblock `mbAddrN` is assigned to `transBlockN`.
 - Otherwise, `transBlockN` is marked as not available.
- Otherwise, if `ctxBlockCat` is equal to 3, the following applies.
 - The derivation process for neighbouring macroblocks specified in subclause 6.4.8.1 is invoked and the output is assigned to `mbAddrN` (with `N` being either `A` or `B`).
 - The variable `transBlockN` is derived as follows.

- If mbAddrN is available, the macroblock mbAddrN does not have mb_type equal to P_Skip, B_Skip, or I_PCM, and CodedBlockPatternChroma is not equal to 0 for the macroblock mbAddrN, the chroma DC block of chroma component iCbCr of macroblock mbAddrN is assigned to transBlockN.
- Otherwise, transBlockN is marked as not available.
- Otherwise (ctxBlockCat is equal to 4), the following applies.
 - The derivation process for neighbouring 4x4 chroma blocks specified in subclause 6.4.8.4 is invoked with chroma4x4BlkIdx as input and the output is assigned to mbAddrN, chroma4x4BlkIdxN (with N being either A or B).
 - The variable transBlockN is derived as follows.
 - If mbAddrN is available, the macroblock mbAddrN does not have mb_type equal to P_Skip, B_Skip, or I_PCM, and CodedBlockPatternChroma is equal to 2 for the macroblock mbAddrN, the 4x4 chroma block with chroma4x4BlkIdxN of the chroma component iCbCr of macroblock mbAddrN is assigned to transBlockN.
 - Otherwise, transBlockN is marked as not available.

Let the variable condTermFlagN (with N being either A or B) be derived as follows.

- If any of the following conditions is true, condTermFlagN is set equal to 0
 - mbAddrN is not available and the current macroblock is coded in Inter prediction mode
 - mbAddrN is available and transBlockN is not available and mb_type for the macroblock mbAddrN is not equal to I_PCM
 - The current macroblock is coded in Intra prediction mode, constrained_intra_pred_flag is equal to 1, the macroblock mbAddrN is available and coded in Inter prediction mode, and slice data partitioning is in use (nal_unit_type is in the range of 2 through 4, inclusive).
- Otherwise, if any of the following conditions is true, condTermFlagN is set equal to 1
 - mbAddrN is not available and the current macroblock is coded in Intra prediction mode
 - mb_type for the macroblock mbAddrN is equal to I_PCM
- Otherwise, condTermFlagN is set equal to the value of the coded_block_flag of the transform block transBlockN that was decoded for the macroblock mbAddrN.

The variable ctxIdxInc(ctxBlockCat) is derived by

$$\text{ctxIdxInc}(\text{ctxBlockCat}) = \text{condTermFlagA} + 2 * \text{condTermFlagB} \quad (9-13)$$

9.3.3.1.1.10 Derivation process of ctxIdxInc for the syntax element transform_size_8x8_flag

Output of this process is ctxIdxInc.

The derivation process for neighbouring macroblocks specified in subclause 6.4.8.1 is invoked and the output is assigned to mbAddrA and mbAddrB.

Let the variable condTermFlagN (with N being either A or B) be derived as follows.

- If any of the following conditions is true, condTermFlagN is set equal to 0.
 - mbAddrN is not available
 - transform_size_8x8_flag for the macroblock mbAddrN is equal to 0
- Otherwise, condTermFlagN is set equal to 1.

The variable ctxIdxInc is derived by

$$\text{ctxIdxInc} = \text{condTermFlagA} + \text{condTermFlagB} \quad (9-14)$$

9.3.3.1.2 Assignment process of ctxIdxInc using prior decoded bin values

Inputs to this process are ctxIdxOffset and binIdx.

Output of this process is ctxIdxInc.

Table 9-32 contains the specification of ctxIdxInc for the given values of ctxIdxOffset and binIdx.

For each value of ctxIdxOffset and binIdx, ctxIdxInc is derived by using some of the values of prior decoded bin values ($b_0, b_1, b_2, \dots, b_k$), where the value of the index k is less than the value of binIdx.

Table 9-32 – Specification of ctxIdxInc for specific values of ctxIdxOffset and binIdx

Value (name) of ctxIdxOffset	binIdx	ctxIdxInc
3	4	$(b_3 \neq 0) ? 5 : 6$
	5	$(b_3 \neq 0) ? 6 : 7$
14	2	$(b_1 \neq 1) ? 2 : 3$
17	4	$(b_3 \neq 0) ? 2 : 3$
27	2	$(b_1 \neq 0) ? 4 : 5$
32	4	$(b_3 \neq 0) ? 2 : 3$
36	2	$(b_1 \neq 0) ? 2 : 3$

9.3.3.1.3 Assignment process of ctxIdxInc for syntax elements significant_coeff_flag, last_significant_coeff_flag, and coeff_abs_level_minus1

Inputs to this process are ctxIdxOffset and binIdx.

Output of this process is ctxIdxInc.

The assignment process of ctxIdxInc for syntax elements significant_coeff_flag, last_significant_coeff_flag, and coeff_abs_level_minus1 as well as for coded_block_flag depends on categories of different blocks denoted by the variable ctxBlockCat. The specification of these block categories is given in Table 9-33.

Table 9-33 – Specification of ctxBlockCat for the different blocks

Block description	maxNumCoeff	ctxBlockCat
block of luma DC transform coefficient levels (i.e., list Intra16x16DCLevel as described in subclause 7.4.5.3)	16	0
block of luma AC transform coefficient levels (i.e., list Intra16x16ACLevel[i] as described in subclause 7.4.5.3)	15	1
block of 16 luma transform coefficient levels (i.e., list LumaLevel[i] as described in subclause 7.4.5.3)	16	2
block of chroma DC transform coefficient levels	$4 * \text{NumC8x8}$	3
block of chroma AC transform coefficient levels	15	4
block of 64 luma transform coefficient levels (i.e., list LumaLevel8x8[i] as described in subclause 7.4.5.3)	64	5

Let the variable levelListIdx be set equal to the index of the list of transform coefficient levels as specified in subclause 7.4.5.3.

For the syntax elements significant_coeff_flag and last_significant_coeff_flag in blocks with ctxBlockCat < 5 and ctxBlockCat != 3, the variable ctxIdxInc is derived by

$$\text{ctxIdxInc} = \text{levelListIdx} \quad (9-15)$$

where levelListIdx ranges from 0 to maxNumCoeff – 2, inclusive.

For the syntax elements significant_coeff_flag and last_significant_coeff_flag in blocks with ctxBlockCat == 3, the variable ctxIdxInc is derived by

$$\text{ctxIdxInc} = \text{Min}(\text{levelListIdx} / \text{NumC8x8}, 2) \quad (9-16)$$

where levelListIdx ranges from 0 to $4 * \text{NumC8x8} - 2$, inclusive.

For the syntax elements `significant_coeff_flag` and `last_significant_coeff_flag` in 8x8 luma blocks with `ctxBlockCat == 5`, Table 9-34 contains the specification of `ctxIdxInc` for the given values of `levelListIdx`, where `levelListIdx` ranges from 0 to 62, inclusive.

Table 9-34 – Mapping of scanning position to `ctxIdxInc` for `ctxBlockCat == 5`

levelListIdx	ctxIdxInc for significant_coeff_flag (frame coded macroblocks)	ctxIdxInc for significant_coeff_flag (field coded macroblocks)	ctxIdxInc for last_significant_coeff_flag	levelListIdx	ctxIdxInc for significant_coeff_flag (frame coded macroblocks)	ctxIdxInc for significant_coeff_flag (field coded macroblocks)	ctxIdxInc for last_significant_coeff_flag
0	0	0	0	32	7	9	3
1	1	1	1	33	6	9	3
2	2	1	1	34	11	10	3
3	3	2	1	35	12	10	3
4	4	2	1	36	13	8	3
5	5	3	1	37	11	11	3
6	5	3	1	38	6	12	3
7	4	4	1	39	7	11	3
8	4	5	1	40	8	9	4
9	3	6	1	41	9	9	4
10	3	7	1	42	14	10	4
11	4	7	1	43	10	10	4
12	4	7	1	44	9	8	4
13	4	8	1	45	8	13	4
14	5	4	1	46	6	13	4
15	5	5	1	47	11	9	4
16	4	6	2	48	12	9	5
17	4	9	2	49	13	10	5
18	4	10	2	50	11	10	5
19	4	10	2	51	6	8	5
20	3	8	2	52	9	13	6
21	3	11	2	53	14	13	6
22	6	12	2	54	10	9	6
23	7	11	2	55	9	9	6
24	7	9	2	56	11	10	7

levelIdx	ctxIdxInc for significant_coeff_flag (frame coded macroblocks)	ctxIdxInc for significant_coeff_flag (field coded macroblocks)	ctxIdxInc for last_significant_coeff_flag	levelIdx	ctxIdxInc for significant_coeff_flag (frame coded macroblocks)	ctxIdxInc for significant_coeff_flag (field coded macroblocks)	ctxIdxInc for last_significant_coeff_flag
25	7	9	2	57	12	10	7
26	8	10	2	58	13	14	7
27	9	10	2	59	11	14	7
28	10	8	2	60	14	14	8
29	9	11	2	61	10	14	8
30	8	12	2	62	12	14	8
31	7	11	2				

Let numDecodAbsLevelEq1 denotes the accumulated number of decoded transform coefficient levels with absolute value equal to 1, and let numDecodAbsLevelGt1 denotes the accumulated number of decoded transform coefficient levels with absolute value greater than 1. Both numbers are related to the same transform coefficient block, where the current decoding process takes place. Then, for decoding of coeff_abs_level_minus1, ctxIdxInc for coeff_abs_level_minus1 is specified depending on binIdx as follows.

- If binIdx is equal to 0, ctxIdxInc is derived by

$$\text{ctxIdxInc} = ((\text{numDecodAbsLevelGt1} \neq 0) ? 0 : \text{Min}(4, 1 + \text{numDecodAbsLevelEq1})) \quad (9-17)$$

- Otherwise (binIdx is greater than 0), ctxIdxInc is derived by

$$\text{ctxIdxInc} = 5 + \text{Min}(4 - (\text{ctxBlockCat} == 3), \text{numDecodAbsLevelGt1}) \quad (9-18)$$

9.3.3.2 Arithmetic decoding process

Inputs to this process are the bypassFlag, ctxIdx as derived in subclause 9.3.3.1, and the state variables codIRange and codIOffset of the arithmetic decoding engine.

Output of this process is the value of the bin.

Figure 9-2 illustrates the whole arithmetic decoding process for a single bin. For decoding the value of a bin, the context index ctxIdx is passed to the arithmetic decoding process DecodeBin(ctxIdx), which is specified as follows.

- If bypassFlag is equal to 1, DecodeBypass() as specified in subclause 9.3.3.2.3 is invoked.
- Otherwise, if bypassFlag is equal to 0 and ctxIdx is equal to 276, DecodeTerminate() as specified in subclause 9.3.3.2.4 is invoked.
- Otherwise (bypassFlag is equal to 0 and ctxIdx is not equal to 276), DecodeDecision() as specified in subclause 9.3.3.2.1 is applied.

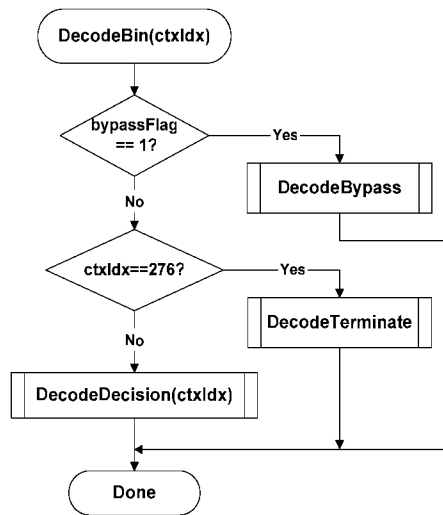


Figure 9-2 – Overview of the arithmetic decoding process for a single bin (informative)

NOTE – Arithmetic coding is based on the principle of recursive interval subdivision. Given a probability estimation $p(0)$ and $p(1) = 1 - p(0)$ of a binary decision $(0, 1)$, an initially given code sub-interval with the range codIRange will be subdivided into two sub-intervals having range $p(0) * \text{codIRange}$ and $\text{codIRange} - p(0) * \text{codIRange}$, respectively. Depending on the decision, which has been observed, the corresponding sub-interval will be chosen as the new code interval, and a binary code string pointing into that interval will represent the sequence of observed binary decisions. It is useful to distinguish between the most probable symbol (MPS) and the least probable symbol (LPS), so that binary decisions have to be identified as either MPS or LPS, rather than 0 or 1. Given this terminology, each context is specified by the probability p_{LPS} of the LPS and the value of MPS (valMPS), which is either 0 or 1.

The arithmetic core engine in this Recommendation | International Standard has three distinct properties:

- The probability estimation is performed by means of a finite-state machine with a table-based transition process between 64 different representative probability states $\{p_{\text{LPS}}(p\text{StateIdx}) \mid 0 \leq p\text{StateIdx} < 64\}$ for the LPS probability p_{LPS} . The numbering of the states is arranged in such a way that the probability state with index $p\text{StateIdx} = 0$ corresponds to an LPS probability value of 0.5, with decreasing LPS probability towards higher state indices.
- The range codIRange representing the state of the coding engine is quantised to a small set $\{Q_1, \dots, Q_4\}$ of pre-set quantisation values prior to the calculation of the new interval range. Storing a table containing all 64×4 pre-computed product values of $Q_i * p_{\text{LPS}}(p\text{StateIdx})$ allows a multiplication-free approximation of the product $\text{codIRange} * p_{\text{LPS}}(p\text{StateIdx})$.
- For syntax elements or parts thereof for which an approximately uniform probability distribution is assumed to be given a separate simplified encoding and decoding bypass process is used.

9.3.3.2.1 Arithmetic decoding process for a binary decision

Inputs to this process are ctxIdx , codIRange , and codIOffset .

Outputs of this process are the decoded value binVal , and the updated variables codIRange and codIOffset .

Figure 9-3 shows the flowchart for decoding a single decision (DecodeDecision).

1. The value of the variable codIRangeLPS is derived as follows.
 - Given the current value of codIRange , the variable qCodIRangeIdx is derived by

$$\text{qCodIRangeIdx} = (\text{codIRange} \gg 6) \& 0x03 \quad (9-19)$$

- Given qCodIRangeIdx and $p\text{StateIdx}$ associated with ctxIdx , the value of the variable rangeTabLPS as specified in Table 9-35 is assigned to codIRangeLPS :

$$\text{codIRangeLPS} = \text{rangeTabLPS}[p\text{StateIdx}][\text{qCodIRangeIdx}] \quad (9-20)$$

2. The variable `codIRange` is set equal to `codIRange – codIRangeLPS` and the following applies.
 - If `codIOffset` is greater than or equal to `codIRange`, the variable `binVal` is set equal to $1 - \text{valMPS}$, `codIOffset` is decremented by `codIRange`, and `codIRange` is set equal to `codIRangeLPS`.
 - Otherwise, the variable `binVal` is set equal to `valMPS`.

Given the value of `binVal`, the state transition is performed as specified in subclause 9.3.3.2.1.1. Depending on the current value of `codIRange`, renormalization is performed as specified in subclause 9.3.3.2.2.

9.3.3.2.1.1 State transition process

Inputs to this process are the current `pStateIdx`, the decoded value `binVal` and `valMPS` values of the context variable associated with `ctxIdx`.

Outputs of this process are the updated `pStateIdx` and `valMPS` of the context variable associated with `ctxIdx`.

Depending on the decoded value `binVal`, the update of the two variables `pStateIdx` and `valMPS` associated with `ctxIdx` is derived as follows:

```

if( binVal == valMPS )
    pStateIdx = transIdxMPS( pStateIdx )
else {
    if( pStateIdx == 0 )
        valMPS = 1 - valMPS
    pStateIdx = transIdxLPS( pStateIdx )
}
  
```

(9-21)

Table 9-36 specifies the transition rules `transIdxMPS()` and `transIdxLPS()` after decoding the value of `valMPS` and $1 - \text{valMPS}$, respectively.

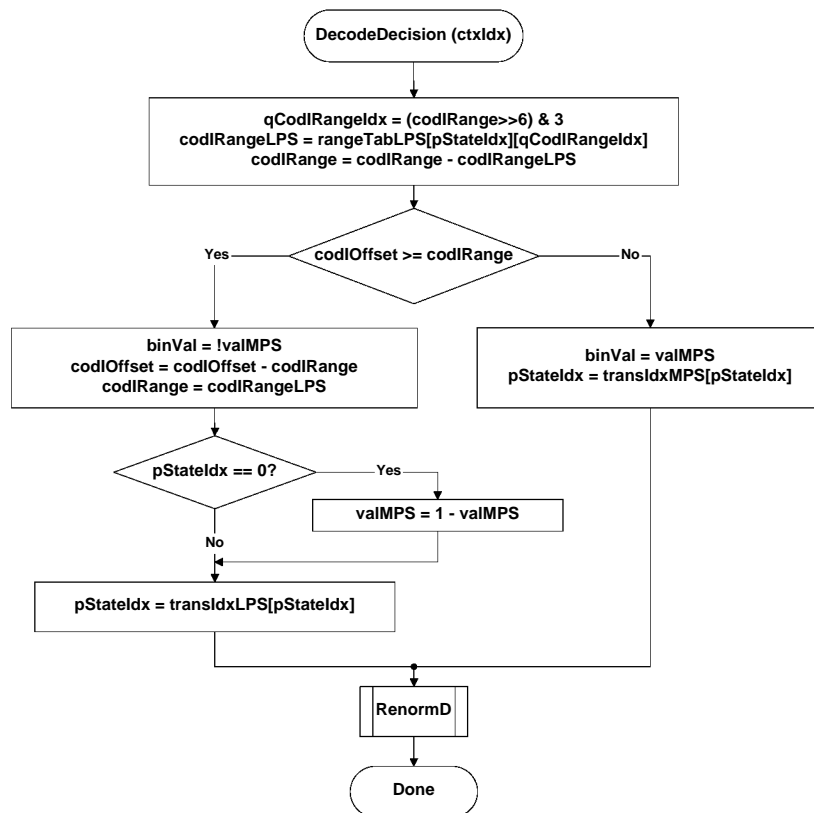


Figure 9-3 – Flowchart for decoding a decision

Table 9-35 – Specification of rangeTabLPS depending on pStateIdx and qCodIRangeIdx

pStateIdx	qCodIRangeIdx				pStateIdx	qCodIRangeIdx			
	0	1	2	3		0	1	2	3
0	128	176	208	240	32	27	33	39	45
1	128	167	197	227	33	26	31	37	43
2	128	158	187	216	34	24	30	35	41
3	123	150	178	205	35	23	28	33	39
4	116	142	169	195	36	22	27	32	37
5	111	135	160	185	37	21	26	30	35
6	105	128	152	175	38	20	24	29	33
7	100	122	144	166	39	19	23	27	31
8	95	116	137	158	40	18	22	26	30
9	90	110	130	150	41	17	21	25	28
10	85	104	123	142	42	16	20	23	27
11	81	99	117	135	43	15	19	22	25
12	77	94	111	128	44	14	18	21	24
13	73	89	105	122	45	14	17	20	23
14	69	85	100	116	46	13	16	19	22
15	66	80	95	110	47	12	15	18	21
16	62	76	90	104	48	12	14	17	20
17	59	72	86	99	49	11	14	16	19
18	56	69	81	94	50	11	13	15	18
19	53	65	77	89	51	10	12	15	17
20	51	62	73	85	52	10	12	14	16
21	48	59	69	80	53	9	11	13	15
22	46	56	66	76	54	9	11	12	14
23	43	53	63	72	55	8	10	12	14
24	41	50	59	69	56	8	9	11	13
25	39	48	56	65	57	7	9	11	12
26	37	45	54	62	58	7	9	10	12
27	35	43	51	59	59	7	8	10	11
28	33	41	48	56	60	6	8	9	11
29	32	39	46	53	61	6	7	9	10
30	30	37	43	50	62	6	7	8	9
31	29	35	41	48	63	2	2	2	2

Table 9-36 – State transition table

pStateIdx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
transIdxLPS	0	0	1	2	2	4	4	5	6	7	8	9	9	11	11	12
transIdxMPS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
pStateIdx	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
transIdxLPS	13	13	15	15	16	16	18	18	19	19	21	21	22	22	23	24
transIdxMPS	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
pStateIdx	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
transIdxLPS	24	25	26	26	27	27	28	29	29	30	30	30	31	32	32	33
transIdxMPS	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
pStateIdx	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
transIdxLPS	33	33	34	34	35	35	35	36	36	36	37	37	37	38	38	63
transIdxMPS	49	50	51	52	53	54	55	56	57	58	59	60	61	62	62	63

9.3.3.2.2 Renormalization process in the arithmetic decoding engine

Inputs to this process are bits from slice data and the variables codIRange and codIOffset.

Outputs of this process are the updated variables codIRange and codIOffset.

A flowchart of the renormalization is shown in Figure 9-4. The current value of codIRange is first compared to 0x0100 and further steps are specified as follows.

- If codIRange is greater than or equal to 0x0100, no renormalization is needed and the RenormD process is finished;
- Otherwise (codIRange is less than 0x0100), the renormalization loop is entered. Within this loop, the value of codIRange is doubled, i.e., left-shifted by 1 and a single bit is shifted into codIOffset by using read_bits(1).

The bitstream shall not contain data that results in a value of codIOffset being greater than or equal to codIRange upon completion of this process.

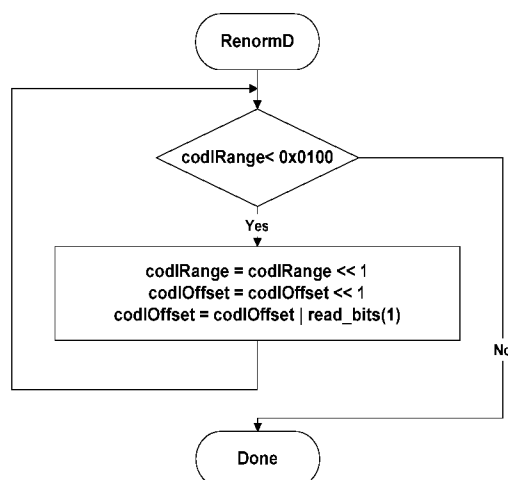


Figure 9-4 – Flowchart of renormalization

9.3.3.2.3 Bypass decoding process for binary decisions

Inputs to this process are bits from slice data and the variables `codIRange` and `codIOffset`.

Outputs of this process are the updated variable `codIOffset` and the decoded value `binVal`.

The bypass decoding process is invoked when `bypassFlag` is equal to 1. Figure 9-5 shows a flowchart of the corresponding process.

First, the value of `codIOffset` is doubled, i.e., left-shifted by 1 and a single bit is shifted into `codIOffset` by using `read_bits(1)`. Then, the value of `codIOffset` is compared to the value of `codIRange` and further steps are specified as follows.

- If `codIOffset` is greater than or equal to `codIRange`, the variable `binVal` is set equal to 1 and `codIOffset` is decremented by `codIRange`.
- Otherwise (`codIOffset` is less than `codIRange`), the variable `binVal` is set equal to 0.

The bitstream shall not contain data that results in a value of `codIOffset` being greater than or equal to `codIRange` upon completion of this process.

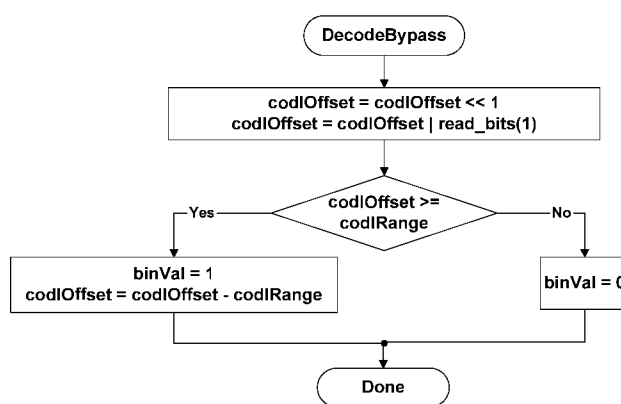


Figure 9-5 – Flowchart of bypass decoding process

9.3.3.2.4 Decoding process for binary decisions before termination

Inputs to this process are bits from slice data and the variables `codIRange` and `codIOffset`.

Outputs of this process are the updated variables `codIRange` and `codIOffset`, and the decoded value `binVal`.

This special decoding routine applies to decoding of `end_of_slice_flag` and of the bin indicating the `L_PCM` mode corresponding to `ctxIdx` equal to 276. Figure 9-6 shows the flowchart of the corresponding decoding process, which is specified as follows.

First, the value of `codIRange` is decremented by 2. Then, the value of `codIOffset` is compared to the value of `codIRange` and further steps are specified as follows.

- If `codIOffset` is greater than or equal to `codIRange`, the variable `binVal` is set equal to 1, no renormalization is carried out, and CABAC decoding is terminated. The last bit inserted in register `codIOffset` is equal to 1. When decoding `end_of_slice_flag`, this last bit inserted in register `codIOffset` is interpreted as `rbsp_stop_one_bit`.
- Otherwise (`codIOffset` is less than `codIRange`), the variable `binVal` is set equal to 0 and renormalization is performed as specified in subclause 9.3.3.2.2.

NOTE – This procedure may also be implemented using `DecodeDecision(ctxIdx)` with `ctxIdx = 276`. In the case where the decoded value is equal to 1, seven more bits would be read by `DecodeDecision(ctxIdx)` and a decoding process would have to adjust its bitstream pointer accordingly to properly decode following syntax elements.

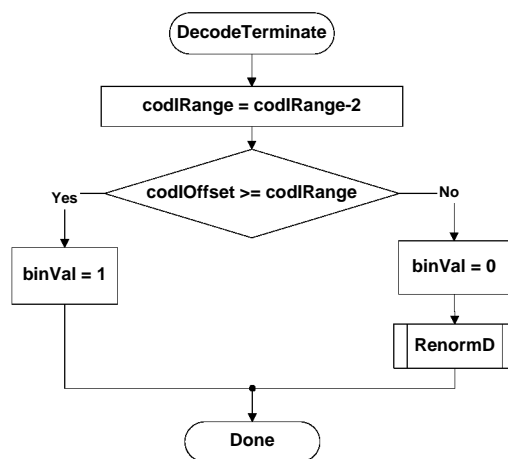


Figure 9-6 – Flowchart of decoding a decision before termination

9.3.4 Arithmetic encoding process (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

Inputs to this process are decisions that are to be encoded and written.

Outputs of this process are bits that are written to the RBSP.

This informative subclause describes an arithmetic encoding engine that matches the arithmetic decoding engine described in subclause 9.3.3.2. The encoding engine is essentially symmetric with the decoding engine, i.e., procedures are called in the same order. The following procedures are described in this section: InitEncoder, EncodeDecision, EncodeBypass, EncodeTerminate, which correspond to InitDecoder, DecodeDecision, DecodeBypass, and DecodeTerminate, respectively. The state of the arithmetic encoding engine is represented by a value of the variable codILow pointing to the lower end of a sub-interval and a value of the variable codIRange specifying the corresponding range of that sub-interval.

9.3.4.1 Initialisation process for the arithmetic encoding engine (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

This process is invoked before encoding the first macroblock of a slice, and after encoding any pcm_alignment_zero_bit and all pcm_sample_luma and pcm_sample_chroma data for a macroblock of type I_PCM.

Outputs of this process are the values codILow, codIRange, firstBitFlag, bitsOutstanding, and symCnt of the arithmetic encoding engine.

In the initialisation procedure of the encoder, codILow is set equal to 0, and codIRange is set equal to 0x01FE. Furthermore, a firstBitFlag is set equal to 1, and bitsOutstanding and symCnt counters are set equal to 0.

NOTE – The minimum register precision required for codILow is 10 bits and for CodIRange is 9 bits. The precision required for the counters bitsOutstanding and symCnt should be sufficiently large to prevent overflow of the related registers. When MaxBinCountInSlice denotes the maximum total number of binary decisions to encode in one slice, the minimum register precision required for the variables bitsOutstanding and symCnt is given by $\text{Ceil}(\text{Log}_2(\text{MaxBinCountInSlice} + 1))$.

9.3.4.2 Encoding process for a binary decision (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

Inputs to this process are the context index ctxIdx, the value of binVal to be encoded, and the variables codIRange, codILow and symCnt.

Outputs of this process are the variables codIRange, codILow, and symCnt.

Figure 9-7 shows the flowchart for encoding a single decision. In a first step, the variable codIRangeLPS is derived as follows.

Given the current value of codIRange, codIRange is mapped to the index qCodIRangeIdx of a quantised value of codIRange by using Equation 9-19. The value of qCodIRangeIdx and the value of pStateIdx associated with ctxIdx are

used to determine the value of the variable rangeTabLPS as specified in Table 9-35, which is assigned to codIRangeLPS. The value of codIRange – codIRangeLPS is assigned to codIRange.

In a second step, the value of binVal is compared to valMPS associated with ctxIdx. When binVal is different from valMPS, codIRange is added to codILow and codIRange is set equal to the value codIRangeLPS. Given the encoded decision, the state transition is performed as specified in subclause 9.3.3.2.1.1. Depending on the current value of codIRange, renormalization is performed as specified in subclause 9.3.4.3. Finally, the variable symCnt is incremented by 1.

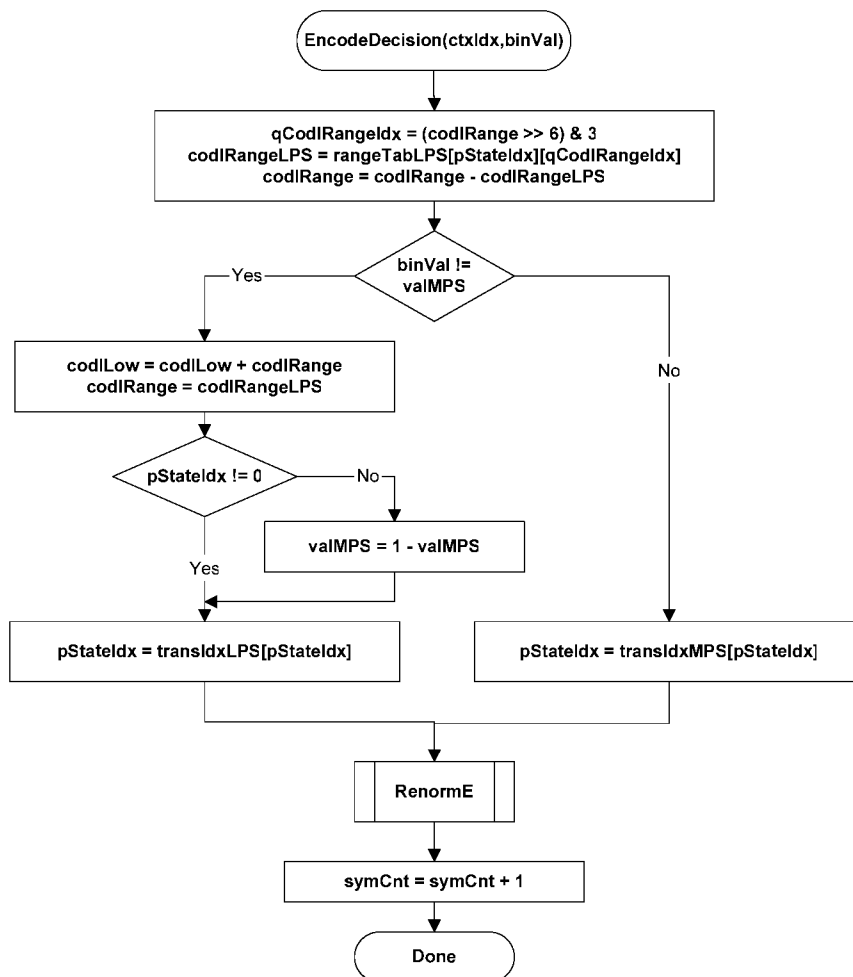


Figure 9-7 – Flowchart for encoding a decision

9.3.4.3 Renormalization process in the arithmetic encoding engine (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

Inputs to this process are the variables codIRange, codILow, firstBitFlag, and bitsOutstanding.

Outputs of this process are zero or more bits written to the RBSP and the updated variables codIRange, codILow, firstBitFlag, and bitsOutstanding.

Renormalization is illustrated in Figure 9-8.

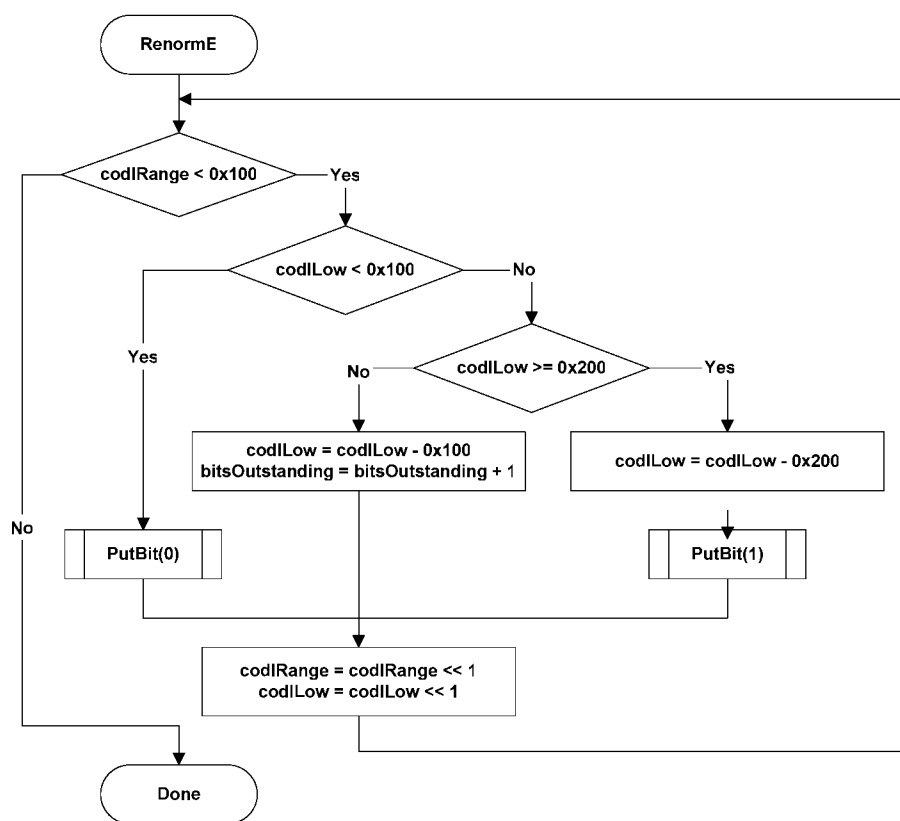


Figure 9-8 – Flowchart of renormalization in the encoder

The PutBit() procedure described in Figure 9-9 provides carry over control. It uses the function WriteBits(B, N) that writes N bits with value B to the bitstream and advances the bitstream pointer by N bit positions. This function assumes the existence of a bitstream pointer with an indication of the position of the next bit to be written to the bitstream by the encoding process.

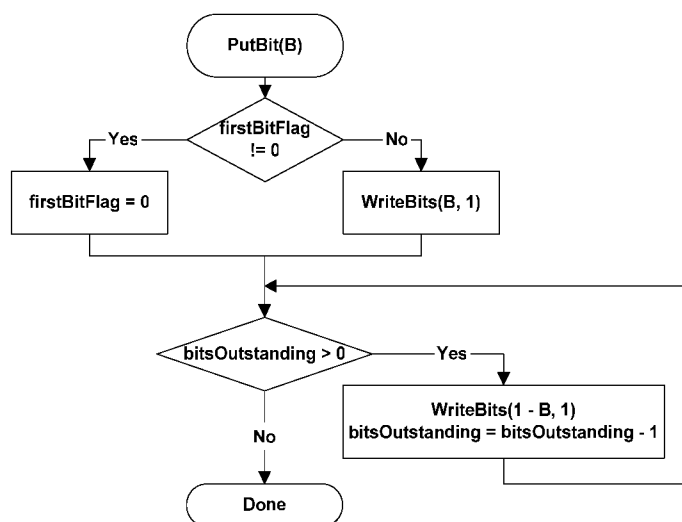


Figure 9-9 – Flowchart of PutBit(B)

9.3.4.4 Bypass encoding process for binary decisions (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

Inputs to this process are the variables `binVal`, `codILow`, `codIRange`, `bitsOutstanding`, and `symCnt`.

Output of this process is a bit written to the RBSP and the updated variables `codILow`, `bitsOutstanding`, and `symCnt`.

This encoding process applies to all binary decisions with `bypassFlag` equal to 1. Renormalization is included in the specification of this process as given in Figure 9-10.

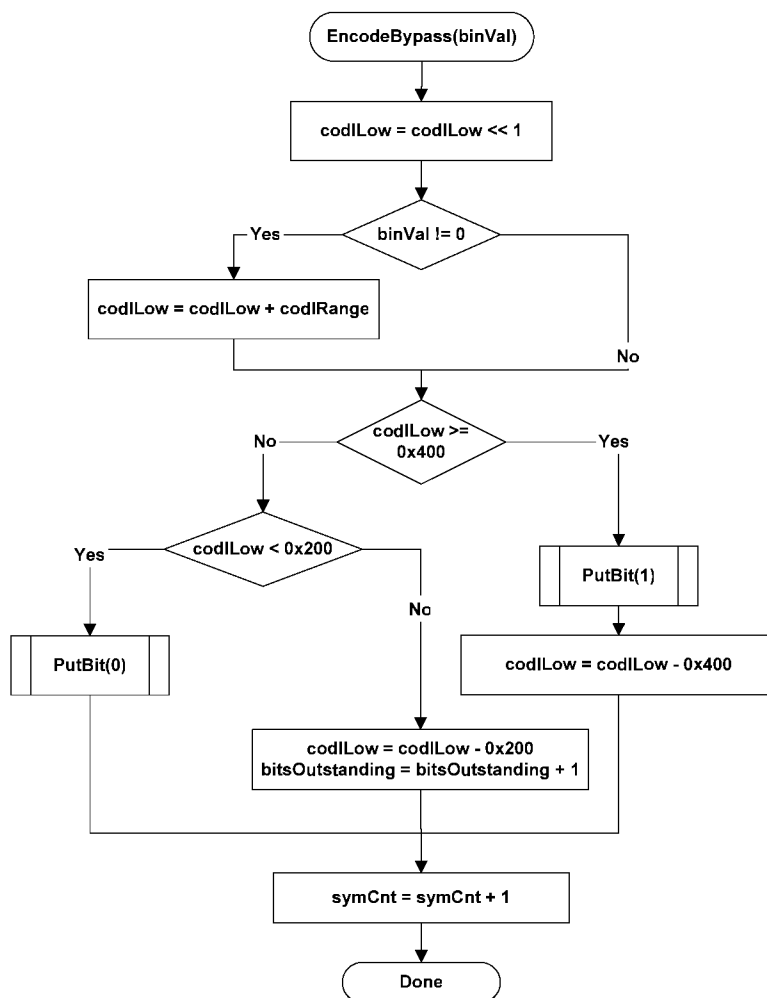


Figure 9-10 – Flowchart of encoding bypass

9.3.4.5 Encoding process for a binary decision before termination (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

Inputs to this process are the variables `binVal`, `codIRange`, `codILow`, `bitsOutstanding`, and `symCnt`.

Outputs of this process are zero or more bits written to the RBSP and the updated variables `codILow`, `codIRange`, `bitsOutstanding`, and `symCnt`.

This encoding routine shown in Figure 9-11 applies to encoding of the `end_of_slice_flag` and of the bin indicating the `L_PCM mb_type` both associated with `ctxIdx` equal to 276.

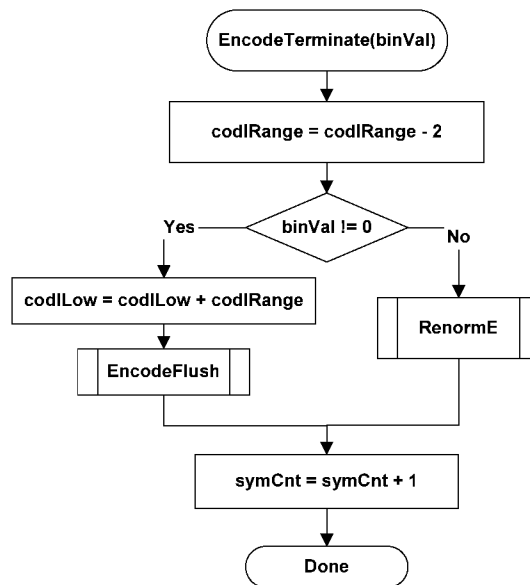


Figure 9-11 – Flowchart of encoding a decision before termination

When the value of binVal to encode is equal to 1, CABAC encoding is terminated and the flushing procedure shown in Figure 9-12 is applied. In this flushing procedure, the last bit written by WriteBits(B, N) is equal to 1. When encoding end_of_slice_flag, this last bit is interpreted as the rbsp_stop_one_bit.

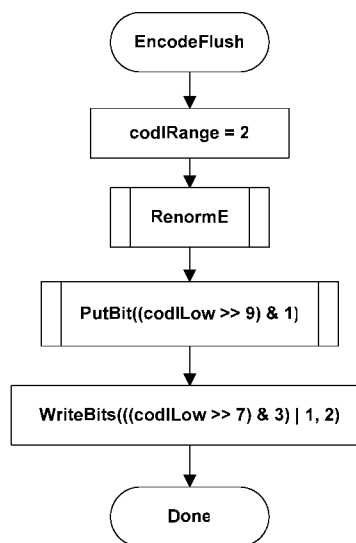


Figure 9-12 – Flowchart of flushing at termination

9.3.4.6 Byte stuffing process (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

This process is invoked after encoding the last macroblock of the last slice of a picture and after encapsulation.

Inputs to this process are the number of bytes NumBytesInVclNALunits of all VCL NAL units of a picture, the number of macroblocks PicSizeInMbs in the picture, and the number of binary symbols BinCountsInNALunits resulting from encoding the contents of all VCL NAL units of the picture.

Outputs of this process are zero or more bytes appended to the NAL unit.

Let the variable k be set equal to $\text{Ceil}((\text{Ceil}(3 * (32 * \text{BinCountsInNALunits} - \text{RawMbBits} * \text{PicSizeInMbs}) \div 1024) - \text{NumBytesInVclNALunits}) \div 3)$. Depending on the variable k the following applies.

- If k is less than or equal to 0, no `cabac_zero_word` is appended to the NAL unit.
- Otherwise (k is greater than 0), the 3-byte sequence `0x000003` is appended k times to the NAL unit after encapsulation, where the first two bytes `0x0000` represent a `cabac_zero_word` and the third byte `0x03` represents an `emulation_prevention_three_byte`.

Annex A

Profiles and levels

(This annex forms an integral part of this Recommendation | International Standard)

Profiles and levels specify restrictions on bitstreams and hence limits on the capabilities needed to decode the bitstreams. Profiles and levels may also be used to indicate interoperability points between individual decoder implementations.

NOTE 1 – This Recommendation | International Standard does not include individually selectable “options” at the decoder, as this would increase interoperability difficulties.

Each profile specifies a subset of algorithmic features and limits that shall be supported by all decoders conforming to that profile.

NOTE 2 – Encoders are not required to make use of any particular subset of features supported in a profile.

Each level specifies a set of limits on the values that may be taken by the syntax elements of this Recommendation | International Standard. The same set of level definitions is used with all profiles, but individual implementations may support a different level for each supported profile. For any given profile, levels generally correspond to decoder processing load and memory capability.

A.1 Requirements on video decoder capability

Capabilities of video decoders conforming to this Recommendation | International Standard are specified in terms of the ability to decode video streams conforming to the constraints of profiles and levels specified in this Annex. For each such profile, the level supported for that profile shall also be expressed.

Specific values are specified in this annex for the syntax elements `profile_idc` and `level_idc`. All other values of `profile_idc` and `level_idc` are reserved for future use by ITU-T | ISO/IEC.

NOTE – Decoders should not infer that when a reserved value of `profile_idc` or `level_idc` falls between the values specified in this Recommendation | International Standard that this indicates intermediate capabilities between the specified profiles or levels, as there are no restrictions on the method to be chosen by ITU-T | ISO/IEC for the use of such future reserved values.

A.2 Profiles

A.2.1 Baseline profile

Bitstreams conforming to the Baseline profile shall obey the following constraints:

- Only I and P slice types may be present.
- NAL unit streams shall not contain `nal_unit_type` values in the range of 2 to 4, inclusive.
- Sequence parameter sets shall have `frame_mbs_only_flag` equal to 1.
- The syntax elements `chroma_format_idc`, `bit_depth_luma_minus8`, `bit_depth_chroma_minus8`, `qpprime_y_zero_transform_bypass_flag`, and `seq_scaling_matrix_present_flag` shall not be present in sequence parameter sets.
- Picture parameter sets shall have `weighted_pred_flag` and `weighted_bipred_idc` both equal to 0.
- Picture parameter sets shall have `entropy_coding_mode_flag` equal to 0.
- Picture parameter sets shall have `num_slice_groups_minus1` in the range of 0 to 7, inclusive.
- The syntax elements `transform_8x8_mode_flag`, `pic_scaling_matrix_present_flag`, and `second_chroma_qp_index_offset` shall not be present in picture parameter sets.
- The syntax element `level_prefix` shall not be greater than 15.
- The level constraints specified for the Baseline profile in subclause A.3 shall be fulfilled.

Conformance of a bitstream to the Baseline profile is specified by `profile_idc` being equal to 66.

Decoders conforming to the Baseline profile at a specific level shall be capable of decoding all bitstreams in which `profile_idc` is equal to 66 or `constraint_set0_flag` is equal to 1 and in which `level_idc` and `constraint_set3_flag` represent a level less than or equal to the specified level.

A.2.2 Main profile

Bitstreams conforming to the Main profile shall obey the following constraints:

- Only I, P, and B slice types may be present.
- NAL unit streams shall not contain `nal_unit_type` values in the range of 2 to 4, inclusive.
- Arbitrary slice order is not allowed.
- The syntax elements `chroma_format_idc`, `bit_depth_luma_minus8`, `bit_depth_chroma_minus8`, `qp_prime_y_zero_transform_bypass_flag`, and `seq_scaling_matrix_present_flag` shall not be present in sequence parameter sets.
- Picture parameter sets shall have `num_slice_groups_minus1` equal to 0 only.
- Picture parameter sets shall have `redundant_pic_cnt_present_flag` equal to 0 only.
- The syntax elements `transform_8x8_mode_flag`, `pic_scaling_matrix_present_flag`, and `second_chroma_qp_index_offset` shall not be present in picture parameter sets.
- The syntax element `level_prefix` shall not be greater than 15 (when present).
- The level constraints specified for the Main profile in subclause A.3 shall be fulfilled.

Conformance of a bitstream to the Main profile is specified by `profile_idc` being equal to 77.

Decoders conforming to the Main profile at a specified level shall be capable of decoding all bitstreams in which `profile_idc` is equal to 77 or `constraint_set1_flag` is equal to 1 and in which `level_idc` and `constraint_set3_flag` represent a level less than or equal to the specified level.

A.2.3 Extended profile

Bitstreams conforming to the Extended profile shall obey the following constraints:

- Sequence parameter sets shall have `direct_8x8_inference_flag` equal to 1.
- The syntax elements `chroma_format_idc`, `bit_depth_luma_minus8`, `bit_depth_chroma_minus8`, `qp_prime_y_zero_transform_bypass_flag`, and `seq_scaling_matrix_present_flag` shall not be present in sequence parameter sets.
- Picture parameter sets shall have `entropy_coding_mode_flag` equal to 0.
- Picture parameter sets shall have `num_slice_groups_minus1` in the range of 0 to 7, inclusive.
- The syntax elements `transform_8x8_mode_flag`, `pic_scaling_matrix_present_flag`, and `second_chroma_qp_index_offset` shall not be present in picture parameter sets.
- The syntax element `level_prefix` shall not be greater than 15 (when present).
- The level constraints specified for the Extended profile in subclause A.3 shall be fulfilled.

Conformance of a bitstream to the Extended profile is specified by `profile_idc` being equal to 88.

Decoders conforming to the Extended profile at a specified level shall be capable of decoding all bitstreams in which `profile_idc` is equal to 88 or `constraint_set2_flag` is equal to 1 and in which `level_idc` represents a level less than or equal to specified level.

Decoders conforming to the Extended profile at a specified level shall also be capable of decoding all bitstreams in which `profile_idc` is equal to 66 or `constraint_set0_flag` is equal to 1, in which `level_idc` and `constraint_set3_flag` represent a level less than or equal to the specified level.

A.2.4 High profile

Bitstreams conforming to the High profile shall obey the following constraints:

- Only I, P, and B slice types may be present.
- NAL unit streams shall not contain `nal_unit_type` values in the range of 2 to 4, inclusive.
- Arbitrary slice order is not allowed.
- Picture parameter sets shall have `num_slice_groups_minus1` equal to 0 only.
- Picture parameter sets shall have `redundant_pic_cnt_present_flag` equal to 0 only.
- Sequence parameter sets shall have `chroma_format_idc` in the range of 0 to 1 inclusive.
- Sequence parameter sets shall have `bit_depth_luma_minus8` equal to 0 only.
- Sequence parameter sets shall have `bit_depth_chroma_minus8` equal to 0 only.

- Sequence parameter sets shall have `qpprime_y_zero_transform_bypass_flag` equal to 0 only.
- The level constraints specified for the High profile in subclause A.3 shall be fulfilled.

Conformance of a bitstream to the High profile is specified by `profile_idc` being equal to 100. Decoders conforming to the High profile at a specific level shall be capable of decoding all bitstreams in which `level_idc` and `constraint_set3_flag` represents a level less than or equal to the specified level and either or both of the following conditions are true:

- `profile_idc` is equal to 77 or 100, or
- `constraint_set1_flag` is equal to 1.

A.2.5 High 10 profile

Bitstreams conforming to the High 10 profile shall obey the following constraints:

- Only I, P, and B slice types may be present.
- NAL unit streams shall not contain `nal_unit_type` values in the range of 2 to 4, inclusive.
- Arbitrary slice order is not allowed.
- Picture parameter sets shall have `num_slice_groups_minus1` equal to 0 only.
- Picture parameter sets shall have `redundant_pic_cnt_present_flag` equal to 0 only.
- Sequence parameter sets shall have `chroma_format_idc` in the range of 0 to 1 inclusive.
- Sequence parameter sets shall have `bit_depth_luma_minus8` in the range of 0 to 2 inclusive.
- Sequence parameter sets shall have `bit_depth_chroma_minus8` in the range of 0 to 2 inclusive.
- Sequence parameter sets shall have `qpprime_y_zero_transform_bypass_flag` equal to 0 only.
- The level constraints specified for the High 10 profile in subclause A.3 shall be fulfilled.

Conformance of a bitstream to the High 10 profile is specified by `profile_idc` being equal to 110. Decoders conforming to the High 10 profile at a specific level shall be capable of decoding all bitstreams in which `level_idc` and `constraint_set3_flag` represent a level less than or equal to the specified level and either or both of the following conditions are true:

- `profile_idc` is equal to 77, 100, or 110, or
- `constraint_set1_flag` is equal to 1.

A.2.6 High 4:2:2 profile

Bitstreams conforming to the High 4:2:2 profile shall obey the following constraints:

- Only I, P, and B slice types may be present.
- NAL unit streams shall not contain `nal_unit_type` values in the range of 2 to 4, inclusive.
- Arbitrary slice order is not allowed.
- Picture parameter sets shall have `num_slice_groups_minus1` equal to 0 only.
- Picture parameter sets shall have `redundant_pic_cnt_present_flag` equal to 0 only.
- Sequence parameter sets shall have `chroma_format_idc` in the range of 0 to 2 inclusive
- Sequence parameter sets shall have `bit_depth_luma_minus8` in the range of 0 to 2 inclusive.
- Sequence parameter sets shall have `bit_depth_chroma_minus8` in the range of 0 to 2 inclusive.
- Sequence parameter sets shall have `qpprime_y_zero_transform_bypass_flag` equal to 0 only.
- The level constraints specified for the High 4:2:2 profile in subclause A.3 shall be fulfilled.

Conformance of a bitstream to the High 4:2:2 profile is specified by `profile_idc` being equal to 122. Decoders conforming to the High 4:2:2 profile at a specific level shall be capable of decoding all bitstreams in which `level_idc` and `constraint_set3_flag` represents a level less than or equal to the specified level and either or both of the following conditions are true:

- `profile_idc` is equal to 77, 100, 110, or 122, or
- `constraint_set1_flag` is equal to 1.

A.2.7 High 4:4:4 profile

Bitstreams conforming to the High 4:4:4 profile shall obey the following constraints:

- Only I, P, and B slice types may be present.
- NAL unit streams shall not contain nal_unit_type values in the range of 2 to 4, inclusive.
- Arbitrary slice order is not allowed.
- Picture parameter sets shall have num_slice_groups_minus1 equal to 0 only.
- Picture parameter sets shall have redundant_pic_cnt_present_flag equal to 0 only.
- Sequence parameter sets shall have bit_depth_luma_minus8 in the range of 0 to 4 inclusive.
- Sequence parameter sets shall have bit_depth_chroma_minus8 in the range of 0 to 4 inclusive.
- The level constraints specified for the High 4:4:4 profile in subclause A.3 shall be fulfilled.

Conformance of a bitstream to the High 4:4:4 profile is specified by profile_idc being equal to 144. Decoders conforming to the High 4:4:4 profile at a specific level shall be capable of decoding all bitstreams in which level_idc and constraint_set3_flag represent a level less than or equal to the specified level and either or both of the following conditions are true:

- profile_idc is equal to 77, 100, 110, 122, or 144, or
- constraint_set1_flag is equal to 1.

A.3 Levels

The following is specified for expressing the constraints in this Annex.

- Let access unit n be the n-th access unit in decoding order with the first access unit being access unit 0.
- Let picture n be the primary coded picture or the corresponding decoded picture of access unit n.

A.3.1 Level limits common to the Baseline, Main, and Extended profiles

Let the variable fR be derived as follows.

- If picture n is a frame, fR is set equal to $1 \div 172$.
- Otherwise (picture n is a field), fR is set equal to $1 \div (172 * 2)$.

Bitstreams conforming to the Baseline, Main, or Extended profiles at a specified level shall obey the following constraints:

- a) The nominal removal time of access unit n (with $n > 0$) from the CPB as specified in subclause C.1.2, satisfies the constraint that $t_{r,n}(n) - t_r(n-1)$ is greater than or equal to $\text{Max}(\text{PicSizeInMbs} \div \text{MaxMBPS}, fR)$, where MaxMBPS is the value specified in Table A-1 that applies to picture n – 1, and PicSizeInMbs is the number of macroblocks in picture n – 1.
- b) The difference between consecutive output times of pictures from the DPB as specified in subclause C.2.2, satisfies the constraint that $\Delta t_{o,dpb}(n) \geq \text{Max}(\text{PicSizeInMbs} \div \text{MaxMBPS}, fR)$, where MaxMBPS is the value specified in Table A-1 for picture n, and PicSizeInMbs is the number of macroblocks of picture n, provided that picture n is a picture that is output and is not the last picture of the bitstream that is output.
- c) The sum of the NumBytesInNALunit variables for access unit 0 is less than or equal to $384 * (\text{PicSizeInMbs} + \text{MaxMBPS} * (t_r(0) - t_{r,n}(0))) \div \text{MinCR}$, where MaxMBPS and MinCR are the values specified in Table A-1 that apply to picture 0 and PicSizeInMbs is the number of macroblocks in picture 0.
- d) The sum of the NumBytesInNALunit variables for access unit n (with $n > 0$) is less than or equal to $384 * \text{MaxMBPS} * (t_r(n) - t_r(n-1)) \div \text{MinCR}$, where MaxMBPS and MinCR are the values specified in Table A-1 that apply to picture n.
- e) $\text{PicWidthInMbs} * \text{FrameHeightInMbs} \leq \text{MaxFS}$, where MaxFS is specified in Table A-1
- f) $\text{PicWidthInMbs} \leq \text{Sqrt}(\text{MaxFS} * 8)$
- g) $\text{FrameHeightInMbs} \leq \text{Sqrt}(\text{MaxFS} * 8)$
- h) $\text{max_dec_frame_buffering} \leq \text{MaxDpbSize}$, where MaxDpbSize is equal to $\text{Min}(1024 * \text{MaxDPB} / (\text{PicWidthInMbs} * \text{FrameHeightInMbs} * 384), 16)$ and MaxDPB is given in Table A-1 in units of 1024 bytes.

- i) For the VCL HRD parameters, $\text{BitRate}[\text{SchedSelIdx}] \leq 1000 * \text{MaxBR}$ and $\text{CpbSize}[\text{SchedSelIdx}] \leq 1000 * \text{MaxCPB}$ for at least one value of SchedSelIdx, where $\text{BitRate}[\text{SchedSelIdx}]$ is given by Equation E-37 and $\text{CpbSize}[\text{SchedSelIdx}]$ is given by Equation E-38 when `vcl_hrd_parameters_present_flag` is equal to 1. MaxBR and MaxCPB are specified in Table A-1 in units of 1000 bits/s and 1000 bits, respectively. The bitstream shall satisfy these conditions for at least one value of SchedSelIdx in the range 0 to `cpb_cnt_minus1`, inclusive.
- j) For the NAL HRD parameters, $\text{BitRate}[\text{SchedSelIdx}] \leq 1200 * \text{MaxBR}$ and $\text{CpbSize}[\text{SchedSelIdx}] \leq 1200 * \text{MaxCPB}$ for at least one value of SchedSelIdx, where $\text{BitRate}[\text{SchedSelIdx}]$ is given by Equation E-37 and $\text{CpbSize}[\text{SchedSelIdx}]$ is given by Equation E-38 when `nal_hrd_parameters_present_flag` equal to 1. MaxBR and MaxCPB are specified in Table A-1 in units of 1200 bits/s and 1200 bits, respectively. The bitstream shall satisfy these conditions for at least one value of SchedSelIdx in the range 0 to `cpb_cnt_minus1`.
- k) Vertical motion vector component range luma motion vectors does not exceed MaxVmvR in units of luma frame samples, where MaxVmvR is specified in Table A-1

NOTE 1 – When `chroma_format_idc` is equal to 1 and the current macroblock is a field macroblock, the motion vector component range for chroma motion vectors may exceed MaxVmvR in units of luma frame samples, due to the method of deriving chroma motion vectors as specified in subclause 8.4.1.4.
- l) Horizontal motion vector range does not exceed the range of -2048 to 2047.75, inclusive, in units of luma samples
- m) Number of motion vectors per two consecutive macroblocks in decoding order (also applying to the total from the last macroblock of a slice and the first macroblock of the next slice in decoding order, and in particular also applying to the total from the last macroblock of the last slice of a picture and the first macroblock of the first slice of the next picture in decoding order) does not exceed MaxMvsPer2Mb, where MaxMvsPer2Mb is specified in Table A-1. The number of motion vectors for each macroblock is the value of the variable MvCnt after the completion of the intra or inter prediction process for the macroblock.
- n) Number of bits of `macroblock_layer()` data for any macroblock is not greater than 3200. Depending on `entropy_coding_mode_flag`, the bits of `macroblock_layer()` data are counted as follows.
 - If `entropy_coding_mode_flag` is equal to 0, the number of bits of `macroblock_layer()` data is given by the number of bits in the `macroblock_layer()` syntax structure for a macroblock.
 - Otherwise (`entropy_coding_mode_flag` is equal to 1), the number of bits of `macroblock_layer()` data for a macroblock is given by the number of times `read_bits(1)` is called in subclauses 9.3.3.2.2 and 9.3.3.2.3 when parsing the `macroblock_layer()` associated with the macroblock.

Table A-1 specifies the limits for each level. Entries marked "-" in Table A-1 denote the absence of a corresponding limit. For purposes of comparison of level capabilities, a level shall be considered to be a lower (higher) level than some other level if the level appears nearer to the top (bottom) row of Table A-1 than the other level.

A level to which the bitstream conforms shall be indicated by the syntax elements `level_idc` and `constraint_set3_flag` as follows.

- If `level_idc` is equal to 11 and `constraint_set3_flag` is equal to 1, the indicated level is level 1b.
- Otherwise (`level_idc` is not equal to 11 or `constraint_set3_flag` is not equal to 1), `level_idc` shall be set equal to a value of ten times the level number specified in Table A-1 and `constraint_set3_flag` shall be set equal to 0.

Table A-1 – Level limits

Level number	Max macroblock processing rate MaxMBPS (MB/s)	Max frame size MaxFS (MBs)	Max decoded picture buffer size MaxDPB (1024 bytes for 4:2:0)	Max video bit rate MaxBR (1000 bits/s, 1200 bits/s, cpbBrVclFactor bits/s, or cpbBrNalFactor bits/s)	Max CPB size MaxCPB (1000 bits, 1200 bits, cpbBrVclFactor bits, or cpbBrNalFactor bits)	Vertical MV component range MaxVmvR (luma frame samples)	Min compression ratio MinCR	Max number of motion vectors per two consecutive MBs MaxMvsPer2Mb
1	1 485	99	148.5	64	175	[-64,+63.75]	2	-
1b	1 485	99	148.5	128	350	[-64,+63.75]	2	-
1.1	3 000	396	337.5	192	500	[-128,+127.75]	2	-
1.2	6 000	396	891.0	384	1 000	[-128,+127.75]	2	-
1.3	11 880	396	891.0	768	2 000	[-128,+127.75]	2	-
2	11 880	396	891.0	2 000	2 000	[-128,+127.75]	2	-
2.1	19 800	792	1 782.0	4 000	4 000	[-256,+255.75]	2	-
2.2	20 250	1 620	3 037.5	4 000	4 000	[-256,+255.75]	2	-
3	40 500	1 620	3 037.5	10 000	10 000	[-256,+255.75]	2	32
3.1	108 000	3 600	6 750.0	14 000	14 000	[-512,+511.75]	4	16
3.2	216 000	5 120	7 680.0	20 000	20 000	[-512,+511.75]	4	16
4	245 760	8 192	12 288.0	20 000	25 000	[-512,+511.75]	4	16
4.1	245 760	8 192	12 288.0	50 000	62 500	[-512,+511.75]	2	16
4.2	522 240	8 704	13 056.0	50 000	62 500	[-512,+511.75]	2	16
5	589 824	22 080	41 400.0	135 000	135 000	[-512,+511.75]	2	16
5.1	983 040	36 864	69 120.0	240 000	240 000	[-512,+511.75]	2	16

Levels with non-integer level numbers in Table A-1 are referred to as “intermediate levels”.

NOTE 2 – All levels have the same status, but some applications may choose to use only the integer-numbered levels.

Informative subclause A.3.4 shows the effect of these limits on frame rates for several example picture formats.

A.3.2 Level limits common to the High, High 10, High 4:2:2, and High 4:4:4 profiles

Let the variable fR be derived as follows.

- If picture n is a frame, fR is set equal to $1 \div 172$.
- Otherwise (picture n is a field), fR is set equal to $1 \div (172 * 2)$.

Bitstreams conforming to the High, High 10, High 4:2:2, or High 4:4:4 profiles at a specified level shall obey the following constraints:

- a) The nominal removal time of access unit n (with $n > 0$) from the CPB as specified in subclause C.1.2, satisfies the constraint that $t_{r,n}(n) - t_r(n-1)$ is greater than or equal to $\text{Max}(\text{PicSizeInMbs} \div \text{MaxMBPS}, \text{fR})$, where MaxMBPS is the value specified in Table A-1 that applies to picture n – 1, and PicSizeInMbs is the number of macroblocks in picture n – 1.
- b) The difference between consecutive output times of pictures from the DPB as specified in subclause C.2.2, satisfies the constraint that $\Delta t_{o,dpb}(n) \geq \text{Max}(\text{PicSizeInMbs} \div \text{MaxMBPS}, \text{fR})$, where MaxMBPS is the value specified in Table A-1 for picture n, and PicSizeInMbs is the number of macroblocks of picture n, provided that picture n is a picture that is output and is not the last picture of the bitstream that is output.
- c) $\text{PicWidthInMbs} * \text{FrameHeightInMbs} \leq \text{MaxFS}$, where MaxFS is specified in Table A-1

- d) $\text{PicWidthInMbs} \leq \text{Sqrt}(\text{MaxFS} * 8)$
- e) $\text{FrameHeightInMbs} \leq \text{Sqrt}(\text{MaxFS} * 8)$
- f) $\text{max_dec_frame_buffering} \leq \text{MaxDpbSize}$, where MaxDpbSize is equal to $\text{Min}(1024 * \text{MaxDPB} / (\text{PicWidthInMbs} * \text{FrameHeightInMbs} * 384), 16)$ and MaxDPB is specified in Table A-1.
- g) Vertical motion vector component range does not exceed MaxVmvR in units of luma frame samples, where MaxVmvR is specified in Table A-1.
- h) Horizontal motion vector range does not exceed the range of -2048 to 2047.75, inclusive, in units of luma samples.
- i) Number of motion vectors per two consecutive macroblocks in decoding order (also applying to the total from the last macroblock of a slice and the first macroblock of the next slice in decoding order) does not exceed MaxMvsPer2Mb , where MaxMvsPer2Mb is specified in Table A-1. The number of motion vectors for each macroblock is value of the variable MvCnt after the completion of the intra or inter prediction process for the macroblock.
- j) Number of bits of $\text{macroblock_layer}()$ data for any macroblock is not greater than $128 + \text{RawMbBits}$. Depending on $\text{entropy_coding_mode_flag}$, the bits of $\text{macroblock_layer}()$ data are counted as follows.
 - If $\text{entropy_coding_mode_flag}$ is equal to 0, the number of bits of $\text{macroblock_layer}()$ data is given by the number of bits in the $\text{macroblock_layer}()$ syntax structure for a macroblock.
 - Otherwise ($\text{entropy_coding_mode_flag}$ is equal to 1), the number of bits of $\text{macroblock_layer}()$ data for a macroblock is given by the number of times $\text{read_bits}(1)$ is called in subclauses 9.3.3.2.2 and 9.3.3.2.3 when parsing the $\text{macroblock_layer}()$ associated with the macroblock.

Table A-1 specifies the limits for each level. Entries marked "-" in Table A-1 denote the absence of a corresponding limit.

A level to which the bitstream conforms shall be indicated by the syntax element level_idc as follows.

- If level_idc is equal to 9, the indicated level is level 1b.
- Otherwise (level_idc is not equal to 9), level_idc shall be set equal to a value of ten times the level number specified in Table A-1.

A.3.3 Profile-specific level limits

- a) In bitstreams conforming to the Main, High, High 10, High 4:2:2, or High 4:4:4 profiles, the removal time of access unit 0 shall satisfy the constraint that the number of slices in picture 0 is less than or equal to $(\text{PicSizeInMbs} + \text{MaxMBPS} * (\text{t}_r(0) - \text{t}_{r,n}(0))) \div \text{SliceRate}$, where SliceRate is the value specified in Table A-4 that applies to picture 0.
- b) In bitstreams conforming to the Main, High, High 10, High 4:2:2, or High 4:4:4 profiles, the difference between consecutive removal time of access units n and $n - 1$ (with $n > 0$) shall satisfy the constraint that the number of slices in picture n is less than or equal to $\text{MaxMBPS} * (\text{t}_r(n) - \text{t}_r(n - 1)) \div \text{SliceRate}$, where SliceRate is the value specified in Table A-4 that applies to picture n .
- c) In bitstreams conforming to the Main, High, High 10, High 4:2:2, or High 4:4:4 profiles, sequence parameter sets shall have $\text{direct_8x8_inference_flag}$ equal to 1 for the levels specified in Table A-4.
 NOTE 1 – $\text{direct_8x8_inference_flag}$ is not relevant to the Baseline profile as it does not allow B slice types (specified in subclause A.2.1), and $\text{direct_8x8_inference_flag}$ is equal to 1 for all levels of the Extended profile (specified in subclause A.2.3).
- d) In bitstreams conforming to the Main, High, High 10, High 4:2:2, or High 4:4:4 or Extended profiles, sequence parameter sets shall have $\text{frame_mbs_only_flag}$ equal to 1 for the levels specified in Table A-4 for the Main, High, High 10, High 4:2:2, and High 4:4:4 profiles and in Table A-5 for the Extended profile.
 NOTE 2 – $\text{frame_mbs_only_flag}$ is equal to 1 for all levels of the Baseline profile (specified in subclause A.2.1).
- e) In bitstreams conforming to the Main, High, High 10, High 4:2:2, or High 4:4:4 or Extended profiles, the value of sub_mb_type in B macroblocks shall not be equal to B_Bi_8x4 , B_Bi_4x8 , or B_Bi_4x4 for the levels in which MinLumaBiPredSize is shown as 8x8 in Table A-4 for the Main, High, High 10, High 4:2:2, and High 4:4:4 profiles and in Table A-5 for the Extended profile.
- f) In bitstreams conforming to the Baseline and Extended profiles, $(\text{xInt}_{\text{max}} - \text{xInt}_{\text{min}} + 6) * (\text{yInt}_{\text{max}} - \text{yInt}_{\text{min}} + 6) \leq \text{MaxSubMbRectSize}$ in macroblocks coded with mb_type equal to P_8x8 , P_8x8ref0 or B_8x8 for all

invocations of the process specified in subclause 8.4.2.2.1 used to generate the predicted luma sample array for a single reference picture list (reference picture list 0 or reference picture list 1) for each 8x8 sub-macroblock, where $\text{NumSubMbPart}(\text{sub_mb_type}) > 1$, where MaxSubMbRectSize is specified in Table A-3 for the Baseline profile and in Table A-5 for the Extended profile and

- xInt_{\min} is the minimum value of xInt_L among all luma sample predictions for the sub-macroblock
 - xInt_{\max} is the maximum value of xInt_L among all luma sample predictions for the sub-macroblock
 - yInt_{\min} is the minimum value of yInt_L among all luma sample predictions for the sub-macroblock
 - yInt_{\max} is the maximum value of yInt_L among all luma sample predictions for the sub-macroblock
- g) In bitstreams conforming to the High, High 10, High 4:2:2, or High 4:4:4 profile, for the VCL HRD parameters, $\text{BitRate}[\text{SchedSelIdx}] \leq \text{cpbBrVclFactor} * \text{MaxBR}$ and $\text{CpbSize}[\text{SchedSelIdx}] \leq \text{cpbBrVclFactor} * \text{MaxCPB}$ for at least one value of SchedSelIdx , where cpbBrVclFactor is specified in Table A-2, $\text{BitRate}[\text{SchedSelIdx}]$ is specified by Equation E-37 and $\text{CpbSize}[\text{SchedSelIdx}]$ is specified by Equation E-38 when $\text{vcl_hrd_parameters_present_flag}$ is equal to 1. MaxBR and MaxCPB are specified in Table A-1 in units of cpbBrVclFactor bits/s and cpbBrVclFactor bits, respectively. The bitstream shall satisfy these conditions for at least one value of SchedSelIdx in the range 0 to cpb_cnt_minus1 , inclusive.
- h) In bitstreams conforming to the High, High 10, High 4:2:2, or High 4:4:4 profile, for the NAL HRD parameters, $\text{BitRate}[\text{SchedSelIdx}] \leq \text{cpbBrNalFactor} * \text{MaxBR}$ and $\text{CpbSize}[\text{SchedSelIdx}] \leq \text{cpbBrNalFactor} * \text{MaxCPB}$ for at least one value of SchedSelIdx , where cpbBrNalFactor is specified in Table A-2, $\text{BitRate}[\text{SchedSelIdx}]$ is specified by Equation E-37 and $\text{CpbSize}[\text{SchedSelIdx}]$ is specified by Equation E-38 when $\text{nal_hrd_parameters_present_flag}$ equal to 1. MaxBR and MaxCPB are specified in Table A-1 in units of cpbBrNalFactor bits/s and cpbBrNalFactor bits, respectively. The bitstream shall satisfy these conditions for at least one value of SchedSelIdx in the range 0 to cpb_cnt_minus1 .

Table A-2 – Specification of cpbBrVclFactor and cpbBrNalFactor

Profile	cpbBrVclFactor	cpbBrNalFactor
High	1 250	1 500
High 10	3 000	3 600
High 4:2:2	4 000	4 800
High 4:4:4	4 000	4 800

A.3.3.1 Baseline profile limits

Table A-3 specifies limits for each level that are specific to bitstreams conforming to the Baseline profile. Entries marked "-" in Table A-3 denote the absence of a corresponding limit.

Table A-3 – Baseline profile level limits

Level number	MaxSubMbRectSize
1	576
1b	576
1.1	576
1.2	576
1.3	576
2	576
2.1	576
2.2	576
3	576
3.1	-
3.2	-
4	-
4.1	-
4.2	-
5	-
5.1	-

A.3.3.2 Main, High, High 10, High 4:2:2, or High 4:4:4 profile limits

Table A-4 specifies limits for each level that are specific to bitstreams conforming to the Main, High, High 10, High 4:2:2, or High 4:4:4 profiles. Entries marked "-" in Table A-4 denote the absence of a corresponding limit.

Table A-4 – Main, High, High 10, High 4:2:2, or High 4:4:4 profile level limits

Level number	SliceRate	MinLumaBiPredSize	direct_8x8_inference_flag	frame_mbs_only_flag
1	-	-	-	1
1b	-	-	-	1
1.1	-	-	-	1
1.2	-	-	-	1
1.3	-	-	-	1
2	-	-	-	1
2.1	-	-	-	-
2.2	-	-	-	-
3	22	-	1	-
3.1	60	8x8	1	-
3.2	60	8x8	1	-
4	60	8x8	1	-
4.1	24	8x8	1	-
4.2	24	8x8	1	1
5	24	8x8	1	1
5.1	24	8x8	1	1

A.3.3.3 Extended Profile Limits

Table A-5 specifies limits for each level that are specific to bitstreams conforming to the Extended profile. Entries marked "-" in Table A-5 denote the absence of a corresponding limit.

Table A-5 – Extended profile level limits

Level number	MaxSubMbRectSize	MinLumaBiPredSize	frame_mbs_only_flag
1	576	-	1
1b	576	-	1
1.1	576	-	1
1.2	576	-	1
1.3	576	-	1
2	576	-	1
2.1	576	-	-
2.2	576	-	-
3	576	-	-
3.1	-	8x8	-
3.2	-	8x8	-
4	-	8x8	-
4.1	-	8x8	-
4.2	-	8x8	1
5	-	8x8	1
5.1	-	8x8	1

A.3.4 Effect of level limits on frame rate (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

Table A-6 – Maximum frame rates (frames per second) for some example frame sizes

Level:					1	1b	1.1	1.2	1.3	2	2.1
Max frame size (macroblocks):					99	99	396	396	396	396	792
Max macroblocks/second:					1 485	1 485	3 000	6 000	11 880	11 880	19 800
Max frame size (samples):					25 344	25 344	101 376	101 376	101 376	101 376	202 752
Max samples/second:					380 160	380 160	768 000	1 536 000	3 041 280	3 041 280	5 068 800
Format	Luma Width	Luma Height	MBs Total	Luma Samples							
SQCIF	128	96	48	12 288	30.9	30.9	62.5	125.0	172.0	172.0	172.0
QCIF	176	144	99	25 344	15.0	15.0	30.3	60.6	120.0	120.0	172.0
QVGA	320	240	300	76 800	-	-	10.0	20.0	39.6	39.6	66.0
525 SIF	352	240	330	84 480	-	-	9.1	18.2	36.0	36.0	60.0
CIF	352	288	396	101 376	-	-	7.6	15.2	30.0	30.0	50.0
525 HHR	352	480	660	168 960	-	-	-	-	-	-	30.0
625 HHR	352	576	792	202 752	-	-	-	-	-	-	25.0
VGA	640	480	1 200	307 200	-	-	-	-	-	-	-
525 4SIF	704	480	1 320	337 920	-	-	-	-	-	-	-
525 SD	720	480	1 350	345 600	-	-	-	-	-	-	-
4CIF	704	576	1 584	405 504	-	-	-	-	-	-	-
625 SD	720	576	1 620	414 720	-	-	-	-	-	-	-
SVGA	800	600	1 900	486 400	-	-	-	-	-	-	-
XGA	1024	768	3 072	786 432	-	-	-	-	-	-	-
720p HD	1280	720	3 600	921 600	-	-	-	-	-	-	-
4VGA	1280	960	4 800	1 228 800	-	-	-	-	-	-	-
SXGA	1280	1024	5 120	1 310 720	-	-	-	-	-	-	-
525 16SIF	1408	960	5 280	1 351 680	-	-	-	-	-	-	-
16CIF	1408	1152	6 336	1 622 016	-	-	-	-	-	-	-
4SVGA	1600	1200	7 500	1 920 000	-	-	-	-	-	-	-
1080 HD	1920	1088	8 160	2 088 960	-	-	-	-	-	-	-
2Kx1K	2048	1024	8 192	2 097 152	-	-	-	-	-	-	-
2Kx1080	2048	1088	8 704	2 228 224	-	-	-	-	-	-	-
4XGA	2048	1536	12 288	3 145 728	-	-	-	-	-	-	-
16VGA	2560	1920	19 200	4 915 200	-	-	-	-	-	-	-
3616x1536 (2.35:1)	3616	1536	21 696	5 554 176	-	-	-	-	-	-	-
3672x1536 (2.39:1)	3680	1536	22 080	5 652 480	-	-	-	-	-	-	-
4Kx2K	4096	2048	32 768	8 388 608	-	-	-	-	-	-	-
4096x2304 (16:9)	4096	2304	36 864	9 437 184	-	-	-	-	-	-	-

Table A-6 (continued) – Maximum frame rates (frames per second) for some example frame sizes

Level:					2.2	3	3.1	3.2	4	4.1	4.2
Max frame size (macroblocks):					1 620	1 620	3 600	5 120	8 192	8 192	8 704
Max macroblocks/second:					20 250	40 500	108 000	216 000	245 760	245 760	522 240
Max frame size (samples):					414 720	414 720	921 600	1 310 720	2 097 152	2 097 152	2 228 224
Max samples/second:					5 184 000	10 368 000	27 648 000	55 296 000	62 914 560	62 914 560	133 693 440
Format	Luma Width	Luma Height	MBs Total	Luma Samples							
SQCIF	128	96	48	12 288	172.0	172.0	172.0	172.0	172.0	172.0	172.0
QCIF	176	144	99	25 344	172.0	172.0	172.0	172.0	172.0	172.0	172.0
QVGA	320	240	300	76 800	67.5	135.0	172.0	172.0	172.0	172.0	172.0
525 SIF	352	240	330	84 480	61.4	122.7	172.0	172.0	172.0	172.0	172.0
CIF	352	288	396	101 376	51.1	102.3	172.0	172.0	172.0	172.0	172.0
525 HHR	352	480	660	168 960	30.7	61.4	163.6	172.0	172.0	172.0	172.0
625 HHR	352	576	792	202 752	25.6	51.1	136.4	172.0	172.0	172.0	172.0
VGA	640	480	1 200	307 200	16.9	33.8	90.0	172.0	172.0	172.0	172.0
525 4SIF	704	480	1 320	337 920	15.3	30.7	81.8	163.6	172.0	172.0	172.0
525 SD	720	480	1 350	345 600	15.0	30.0	80.0	160.0	172.0	172.0	172.0
4CIF	704	576	1 584	405 504	12.8	25.6	68.2	136.4	155.2	155.2	172.0
625 SD	720	576	1 620	414 720	12.5	25.0	66.7	133.3	151.7	151.7	172.0
SVGA	800	600	1 900	486 400	-	-	56.8	113.7	129.3	129.3	172.0
XGA	1024	768	3 072	786 432	-	-	35.2	70.3	80.0	80.0	172.0
720p HD	1280	720	3 600	921 600	-	-	30.0	60.0	68.3	68.3	145.1
4VGA	1280	960	4 800	1 228 800	-	-	-	45.0	51.2	51.2	108.8
SXGA	1280	1024	5 120	1 310 720	-	-	-	42.2	48.0	48.0	102.0
525 16SIF	1408	960	5 280	1 351 680	-	-	-	-	46.5	46.5	98.9
16CIF	1408	1152	6 336	1 622 016	-	-	-	-	38.8	38.8	82.4
4SVGA	1600	1200	7 500	1 920 000	-	-	-	-	32.8	32.8	69.6
1080 HD	1920	1088	8 160	2 088 960	-	-	-	-	30.1	30.1	64.0
2Kx1K	2048	1024	8 192	2 097 152	-	-	-	-	30.0	30.0	63.8
2Kx1080	2048	1088	8 704	2 228 224	-	-	-	-	-	-	60.0
4XGA	2048	1536	12 288	3 145 728	-	-	-	-	-	-	-
16VGA	2560	1920	19 200	4 915 200	-	-	-	-	-	-	-
3616x1536 (2.35:1)	3616	1536	21 696	5 554 176	-	-	-	-	-	-	-
3672x1536 (2.39:1)	3680	1536	22 080	5 652 480	-	-	-	-	-	-	-
4Kx2K	4096	2048	32 768	8 388 608	-	-	-	-	-	-	-
4096x2304 (16:9)	4096	2304	36 864	9 437 184	-	-	-	-	-	-	-

Table A-6 (concluded) – Maximum frame rates (frames per second) for some example frame sizes

Level:					5	5.1
Max frame size (macroblocks):					22 080	36 864
Max macroblocks/second:					589 824	983 040
Max frame size (samples):					5 652 480	9 437 184
Max samples/second:					150 994 944	251 658 240
Format	Luma Width	Luma Height	MBs Total	Luma Samples		
SQCIF	128	96	48	12 288	172.0	172.0
QCIF	176	144	99	25 344	172.0	172.0
QVGA	320	240	300	76 800	172.0	172.0
525 SIF	352	240	330	84 480	172.0	172.0
CIF	352	288	396	101 376	172.0	172.0
525 HHR	352	480	660	168 960	172.0	172.0
625 HHR	352	576	792	202 752	172.0	172.0
VGA	640	480	1 200	307 200	172.0	172.0
525 4SIF	704	480	1 320	337 920	172.0	172.0
525 SD	720	480	1 350	345 600	172.0	172.0
4CIF	704	576	1 584	405 504	172.0	172.0
625 SD	720	576	1 620	414 720	172.0	172.0
SVGA	800	600	1 900	486 400	172.0	172.0
XGA	1024	768	3 072	786 432	172.0	172.0
720p HD	1280	720	3 600	921 600	163.8	172.0
4VGA	1280	960	4 800	1 228 800	122.9	172.0
SXGA	1280	1024	5 120	1 310 720	115.2	172.0
525 16SIF	1408	960	5 280	1 351 680	111.7	172.0
16CIF	1408	1152	6 336	1 622 016	93.1	155.2
4SVGA	1600	1200	7 500	1 920 000	78.6	131.1
1080 HD	1920	1088	8 160	2 088 960	72.3	120.5
2Kx1K	2048	1024	8 192	2 097 152	72.0	120.0
2Kx1080	2048	1088	8 704	2 228 224	67.8	112.9
4XGA	2048	1536	12 288	3 145 728	48.0	80.0
16VGA	2560	1920	19 200	4 915 200	30.7	51.2
3616x1536 (2.35:1)	3616	1536	21 696	5 554 176	27.2	45.3
3672x1536 (2.39:1)	3680	1536	22 080	5 652 480	26.7	44.5
4Kx2K	4096	2048	32 768	8 388 608	-	30.0
4096x2304 (16:9)	4096	2304	36 864	9 437 184	-	26.7

The following should be noted.

- This Recommendation | International Standard is a variable-frame-size specification. The specific frame sizes in Table A-6 are illustrative examples only.
- As used in Table A-6, "525" refers to typical use for environments using 525 analogue scan lines (of which approximately 480 lines contain the visible picture region), and "625" refers to environments using 625 analogue scan lines (of which approximately 576 lines contain the visible picture region).
- XGA is also known as (aka) XVGA, 4SVGA aka UXGA, 16XGA aka 4Kx3K, CIF aka 625 SIF, 625 HHR aka 2CIF aka half 625 D-1, aka half 625 ITU-R BT.601, 525 SD aka 525 D-1 aka 525 ITU-R BT.601, 625 SD aka 625 D-1 aka 625 ITU-R BT.601.
- Frame rates given are correct for progressive scan modes. The frame rates are also correct for interlaced video coding for the cases of frame height divisible by 32.

Annex B

Byte stream format

(This annex forms an integral part of this Recommendation | International Standard)

This annex specifies syntax and semantics of a byte stream format specified for use by applications that deliver some or all of the NAL unit stream as an ordered stream of bytes or bits within which the locations of NAL unit boundaries need to be identifiable from patterns in the data, such as ITU-T Rec. H.222.0 | ISO/IEC 13818-1 systems or ITU-T Rec. H.320 systems. For bit-oriented delivery, the bit order for the byte stream format is specified to start with the MSB of the first byte, proceed to the LSB of the first byte, followed by the MSB of the second byte, etc.

The byte stream format consists of a sequence of byte stream NAL unit syntax structures. Each byte stream NAL unit syntax structure contains one start code prefix followed by one `nal_unit(NumBytesInNALunit)` syntax structure. It may (and under some circumstances, it shall) also contain an additional `zero_byte` syntax element. It may also contain one or more additional `trailing_zero_8bits` syntax elements. When it is the first byte stream NAL unit in the bitstream, it may also contain one or more additional `leading_zero_8bits` syntax elements.

B.1 Byte stream NAL unit syntax and semantics

B.1.1 Byte stream NAL unit syntax

byte_stream_nal_unit(NumBytesInNALunit) {	C	Descriptor
while(next_bits(24) != 0x000001 && next_bits(32) != 0x00000001)		
leading_zero_8bits /* equal to 0x00 */		f(8)
if(next_bits(24) != 0x000001)		
zero_byte /* equal to 0x00 */		f(8)
start_code_prefix_one_3bytes /* equal to 0x000001 */		f(24)
nal_unit(NumBytesInNALunit)		
while(more_data_in_byte_stream() && next_bits(24) != 0x000001 && next_bits(32) != 0x00000001)		
trailing_zero_8bits /* equal to 0x00 */		f(8)
}		

B.1.2 Byte stream NAL unit semantics

The order of byte stream NAL units in the byte stream shall follow the decoding order of the NAL units contained in the byte stream NAL units (see subclause 7.4.1.2). The content of each byte stream NAL unit is associated with the same access unit as the NAL unit contained in the byte stream NAL unit (see subclause 7.4.1.2.3).

leading_zero_8bits is a byte equal to 0x00.

NOTE – The `leading_zero_8bits` syntax element can only be present in the first byte stream NAL unit of the bitstream, because (as shown in the syntax diagram of subclause B.1.1) any bytes equal to 0x00 that follow a NAL unit syntax structure and precede the four-byte sequence 0x00000001 (which is to be interpreted as a `zero_byte` followed by a `start_code_prefix_one_3bytes`) will be considered to be `trailing_zero_8bits` syntax elements that are part of the preceding byte stream NAL unit.

zero_byte is a single byte equal to 0x00.

When any of the following conditions are fulfilled, the `zero_byte` syntax element shall be present.

- the `nal_unit_type` within the `nal_unit()` is equal to 7 (sequence parameter set) or 8 (picture parameter set)
- the byte stream NAL unit syntax structure contains the first NAL unit of an access unit in decoding order, as specified by subclause 7.4.1.2.3.

start_code_prefix_one_3bytes is a fixed-value sequence of 3 bytes equal to 0x000001. This syntax element is called a start code prefix.

trailing_zero_8bits is a byte equal to 0x00.

B.2 Byte stream NAL unit decoding process

Input to this process consists of an ordered stream of bytes consisting of a sequence of byte stream NAL unit syntax structures.

Output of this process consists of a sequence of NAL unit syntax structures.

At the beginning of the decoding process, the decoder initialises its current position in the byte stream to the beginning of the byte stream. It then extracts and discards each `leading_zero_8bits` syntax element (if present), moving the current position in the byte stream forward one byte at a time, until the current position in the byte stream is such that the next four bytes in the bitstream form the four-byte sequence `0x00000001`.

The decoder then performs the following step-wise process repeatedly to extract and decode each NAL unit syntax structure in the byte stream until the end of the byte stream has been encountered (as determined by unspecified means) and the last NAL unit in the byte stream has been decoded:

1. When the next four bytes in the bitstream form the four-byte sequence `0x00000001`, the next byte in the byte stream (which is a `zero_byte` syntax element) is extracted and discarded and the current position in the byte stream is set equal to the position of the byte following this discarded byte.
2. The next three-byte sequence in the byte stream (which is a `start_code_prefix_one_3bytes`) is extracted and discarded and the current position in the byte stream is set equal to the position of the byte following this three-byte sequence.
3. `NumBytesInNALunit` is set equal to the number of bytes starting with the byte at the current position in the byte stream up to and including the last byte that precedes the location of any of the following conditions:
 - a. A subsequent byte-aligned three-byte sequence equal to `0x000000`, or
 - b. A subsequent byte-aligned three-byte sequence equal to `0x000001`, or
 - c. The end of the byte stream, as determined by unspecified means.
4. `NumBytesInNALunit` bytes are removed from the bitstream and the current position in the byte stream is advanced by `NumBytesInNALunit` bytes. This sequence of bytes is `nal_unit(NumBytesInNALunit)` and is decoded using the NAL unit decoding process.
5. When the current position in the byte stream is not at the end of the byte stream (as determined by unspecified means) and the next bytes in the byte stream do not start with a three-byte sequence equal to `0x000001` and the next bytes in the byte stream do not start with a four byte sequence equal to `0x00000001`, the decoder extracts and discards each `trailing_zero_8bits` syntax element, moving the current position in the byte stream forward one byte at a time, until the current position in the byte stream is such that the next bytes in the byte stream form the four-byte sequence `0x00000001` or the end of the byte stream has been encountered (as determined by unspecified means).

B.3 Decoder byte-alignment recovery (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

Many applications provide data to a decoder in a manner that is inherently byte aligned, and thus have no need for the bit-oriented byte alignment detection procedure described in this subclause.

A decoder is said to have byte-alignment with a bitstream when the decoder is able to determine whether or not the positions of data in the bitstream are byte-aligned. When a decoder does not have byte alignment with the encoder's byte stream, the decoder may examine the incoming bitstream for the binary pattern '00000000 00000000 00000000 00000001' (31 consecutive bits equal to 0 followed by a bit equal to 1). The bit immediately following this pattern is the first bit of an aligned byte following a start code prefix. Upon detecting this pattern, the decoder will be byte aligned with the encoder and positioned at the start of a NAL unit in the byte stream.

Once byte aligned with the encoder, the decoder can examine the incoming byte stream for subsequent three-byte sequences `0x000001` and `0x000003`.

When the three-byte sequence `0x000001` is detected, this is a start code prefix.

When the three-byte sequence `0x000003` is detected, the third byte (`0x03`) is an `emulation_prevention_three_byte` to be discarded as specified in subclause 7.4.1.

When an error in the bitstream syntax is detected (e.g., a non-zero value of the `forbidden_zero_bit` or one of the three-byte or four-byte sequences that are prohibited in subclause 7.4.1), the decoder may consider the detected condition as an indication that byte alignment may have been lost and may discard all bitstream data until the detection of byte alignment at a later position in the bitstream as described in this subclause.

Recommendation | International Standard. For the purpose of counting bits, only the appropriate bits that are actually present in the bitstream are counted.

NOTE 1 – As an example, synchronization of a non-VCL NAL unit, conveyed by means other than presence in the bitstream, with the NAL units that are present in the bitstream, can be achieved by indicating two points in the bitstream, between which the non-VCL NAL unit would have been present in the bitstream, had the encoder decided to convey it in the bitstream.

When the content of a non-VCL NAL unit is conveyed for the application by some means other than presence within the bitstream, the representation of the content of the non-VCL NAL unit is not required to use the same syntax specified in this annex.

NOTE 2 – When HRD information is contained within the bitstream, it is possible to verify the conformance of a bitstream to the requirements of this subclause based solely on information contained in the bitstream. When the HRD information is not present in the bitstream, as is the case for all "stand-alone" Type I bitstreams, conformance can only be verified when the HRD data is supplied by some other means not specified in this Recommendation | International Standard.

The HRD contains a coded picture buffer (CPB), an instantaneous decoding process, a decoded picture buffer (DPB), and output cropping as shown in Figure C-2.

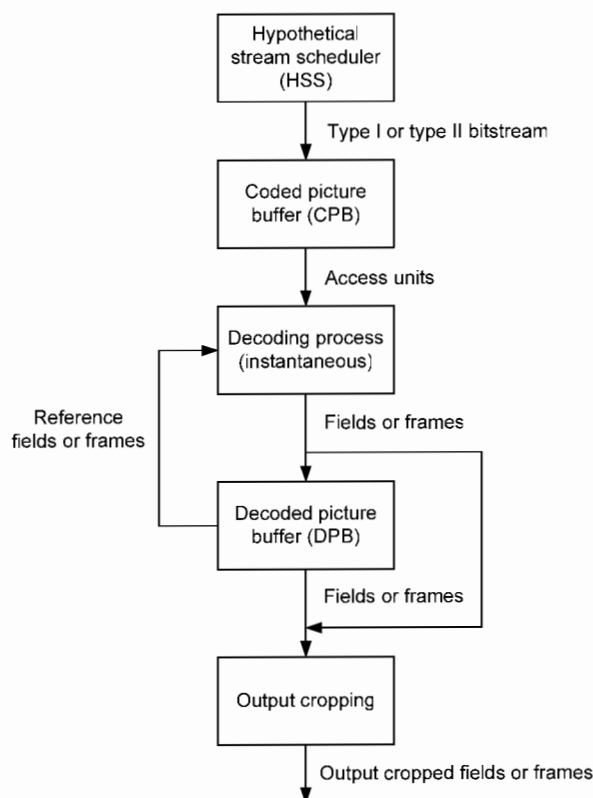


Figure C-2 – HRD buffer model

The CPB size (number of bits) is $CpbSize[SchedSelIdx]$. The DPB size (number of frame buffers) is $Max(1, max_dec_frame_buffering)$.

The HRD operates as follows. Data associated with access units that flow into the CPB according to a specified arrival schedule are delivered by the HSS. The data associated with each access unit are removed and decoded instantaneously by the instantaneous decoding process at CPB removal times. Each decoded picture is placed in the DPB at its CPB removal time unless it is output at its CPB removal time and is a non-reference picture. When a picture is placed in the DPB it is removed from the DPB at the later of the DPB output time or the time that it is marked as "unused for reference".

The operation of the CPB is specified in subclause C.1. The instantaneous decoder operation is specified in clauses 8 and 9. The operation of the DPB is specified in subclause C.2. The output cropping is specified in subclause C.2.2.

HSS and HRD information concerning the number of enumerated delivery schedules and their associated bit rates and buffer sizes is specified in subclauses E.1.1, E.1.2, E.2.1 and E.2.2. The HRD is initialised as specified by the buffering period SEI message as specified in subclauses D.1.1 and D.2.1. The removal timing of access units from the CPB and output timing from the DPB are specified in the picture timing SEI message as specified in subclauses D.1.2 and D.2.2. All timing information relating to a specific access unit shall arrive prior to the CPB removal time of the access unit.

The HRD is used to check conformance of bitstreams and decoders as specified in subclauses C.3 and C.4, respectively.

NOTE 3 – While conformance is guaranteed under the assumption that all frame-rates and clocks used to generate the bitstream match exactly the values signalled in the bitstream, in a real system each of these may vary from the signalled or specified value.

All the arithmetic in this annex is done with real values, so that no rounding errors can propagate. For example, the number of bits in a CPB just prior to or after removal of an access unit is not necessarily an integer.

The variable t_c is derived as follows and is called a clock tick.

$$t_c = \text{num_units_in_tick} \div \text{time_scale} \quad (\text{C-1})$$

The following is specified for expressing the constraints in this Annex.

- Let access unit n be the n -th access unit in decoding order with the first access unit being access unit 0.
- Let picture n be the primary coded picture or the decoded primary picture of access unit n .

C.1 Operation of coded picture buffer (CPB)

The specifications in this subclause apply independently to each set of CPB parameters that is present and to both the Type I and Type II conformance points shown in Figure C-1.

C.1.1 Timing of bitstream arrival

The HRD may be initialised at any one of the buffering period SEI messages. Prior to initialisation, the CPB is empty.

NOTE – After initialisation, the HRD is not initialised again by subsequent buffering period SEI messages.

Each access unit is referred to as access unit n , where the number n identifies the particular access unit. The access unit that is associated with the buffering period SEI message that initializes the CPB is referred to as access unit 0. The value of n is incremented by 1 for each subsequent access unit in decoding order.

The time at which the first bit of access unit n begins to enter the CPB is referred to as the initial arrival time $t_{ai}(n)$.

The initial arrival time of access units is derived as follows.

- If the access unit is access unit 0, $t_{ai}(0) = 0$,
- Otherwise (the access unit is access unit n with $n > 0$), the following applies.
 - If $\text{cbr_flag}[\text{SchedSelIdx}]$ is equal to 1, the initial arrival time for access unit n , is equal to the final arrival time (which is derived below) of access unit $n - 1$, i.e.

$$t_{ai}(n) = t_{af}(n - 1) \quad (\text{C-2})$$

- Otherwise ($\text{cbr_flag}[\text{SchedSelIdx}]$ is equal to 0), the initial arrival time for access unit n is derived by

$$t_{ai}(n) = \text{Max}(t_{af}(n - 1), t_{ai,earliest}(n)) \quad (\text{C-3})$$

where $t_{ai,earliest}(n)$ is derived as follows

- If access unit n is not the first access unit of a subsequent buffering period, $t_{ai,earliest}(n)$ is derived as

$$t_{ai,earliest}(n) = t_{r,n}(n) - (\text{initial_cpb_removal_delay}[\text{SchedSelIdx}] + \text{initial_cpb_removal_delay_offset}[\text{SchedSelIdx}]) \div 90000 \quad (\text{C-4})$$

with $t_{r,n}(n)$ being the nominal removal time of access unit n from the CPB as specified in subclause C.1.2 and $\text{initial_cpb_removal_delay}[\text{SchedSelIdx}]$ and $\text{initial_cpb_removal_delay_offset}[\text{SchedSelIdx}]$ being specified in the previous buffering period SEI message.

- Otherwise (access unit n is the first access unit of a subsequent buffering period), $t_{ai,earliest}(n)$ is derived as

$$t_{ai,earliest}(n) = t_{r,n}(n) - (\text{initial_cpb_removal_delay}[\text{SchedSelIdx}] \div 90000) \quad (\text{C-5})$$

with $\text{initial_cpb_removal_delay}[\text{SchedSelIdx}]$ being specified in the buffering period SEI message associated with access unit n .

The final arrival time for access unit n is derived by

$$t_{af}(n) = t_{ai}(n) + b(n) \div \text{BitRate}[\text{SchedSelIdx}] \quad (\text{C-6})$$

where $b(n)$ is the size in bits of access unit n , counting the bits of the VCL NAL units and the filler data NAL units for the Type I conformance point or all bits of the Type II bitstream for the Type II conformance point, where the Type I and Type II conformance points are as shown in Figure C-1.

The values of SchedSelIdx , $\text{BitRate}[\text{SchedSelIdx}]$, and $\text{CpbSize}[\text{SchedSelIdx}]$ are constrained as follows.

- If access unit n and access unit $n - 1$ are part of different coded video sequences and the content of the active sequence parameter sets of the two coded video sequences differ, the HSS selects a value SchedSelIdx1 of SchedSelIdx from among the values of SchedSelIdx provided for the coded video sequence containing access unit n that results in a $\text{BitRate}[\text{SchedSelIdx1}]$ or $\text{CpbSize}[\text{SchedSelIdx1}]$ for the second of the two coded video sequences (which contains access unit n). The value of $\text{BitRate}[\text{SchedSelIdx1}]$ or $\text{CpbSize}[\text{SchedSelIdx1}]$ may differ from the value of $\text{BitRate}[\text{SchedSelIdx0}]$ or $\text{CpbSize}[\text{SchedSelIdx0}]$ for the value SchedSelIdx0 of SchedSelIdx that was in use for the coded video sequence containing access unit $n - 1$.
- Otherwise, the HSS continues to operate with the previous values of SchedSelIdx , $\text{BitRate}[\text{SchedSelIdx}]$ and $\text{CpbSize}[\text{SchedSelIdx}]$.

When the HSS selects values of $\text{BitRate}[\text{SchedSelIdx}]$ or $\text{CpbSize}[\text{SchedSelIdx}]$ that differ from those of the previous access unit, the following applies.

- the variable $\text{BitRate}[\text{SchedSelIdx}]$ comes into effect at time $t_{ai}(n)$
- the variable $\text{CpbSize}[\text{SchedSelIdx}]$ comes into effect as follows.
 - If the new value of $\text{CpbSize}[\text{SchedSelIdx}]$ exceeds the old CPB size, it comes into effect at time $t_{ai}(n)$,
 - Otherwise, the new value of $\text{CpbSize}[\text{SchedSelIdx}]$ comes into effect at the time $t_r(n)$.

C.1.2 Timing of coded picture removal

For access unit 0, the nominal removal time of the access unit from the CPB is specified by

$$t_{r,n}(0) = \text{initial_cpb_removal_delay}[\text{SchedSelIdx}] \div 90000 \quad (\text{C-7})$$

For the first access unit of a buffering period that does not initialise the HRD, the nominal removal time of the access unit from the CPB is specified by

$$t_{r,n}(n) = t_{r,n}(n_b) + t_c * \text{cpb_removal_delay}(n) \quad (\text{C-8})$$

where $t_{r,n}(n_b)$ is the nominal removal time of the first access unit of the previous buffering period and $\text{cpb_removal_delay}(n)$ is the value of cpb_removal_delay specified in the picture timing SEI message associated with access unit n .

When an access unit n is the first access unit of a buffering period, n_b is set equal to n at the removal time of access unit n .

The nominal removal time $t_{r,n}(n)$ of an access unit n that is not the first access unit of a buffering period is given by

$$t_{r,n}(n) = t_{r,n}(n_b) + t_c * \text{cpb_removal_delay}(n) \quad (\text{C-9})$$

where $t_{r,n}(n_b)$ is the nominal removal time of the first access unit of the current buffering period and $\text{cpb_removal_delay}(n)$ is the value of cpb_removal_delay specified in the picture timing SEI message associated with access unit n .

The removal time of access unit n is specified as follows.

- If `low_delay_hrd_flag` is equal to 0 or $t_{r,n}(n) \geq t_{af}(n)$, the removal time of access unit n is specified by

$$t_r(n) = t_{r,n}(n) \quad (C-10)$$

- Otherwise (`low_delay_hrd_flag` is equal to 1 and $t_{r,n}(n) < t_{af}(n)$), the removal time of access unit n is specified by

$$t_r(n) = t_{r,n}(n) + t_c * \text{Ceil}((t_{af}(n) - t_{r,n}(n)) \div t_c) \quad (C-11)$$

NOTE – The latter case indicates that the size of access unit n , $b(n)$, is so large that it prevents removal at the nominal removal time.

C.2 Operation of the decoded picture buffer (DPB)

The decoded picture buffer contains frame buffers. Each of the frame buffers may contain a decoded frame, a decoded complementary field pair or a single (non-paired) decoded field that are marked as "used for reference" (reference pictures) or are held for future output (reordered or delayed pictures). Prior to initialisation, the DPB is empty (the DPB fullness is set to zero). The following steps of the subclauses of this subclause all happen instantaneously at $t_r(n)$ and in the sequence listed.

C.2.1 Decoding of gaps in `frame_num` and storage of "non-existing" frames

If applicable, gaps in `frame_num` are detected by the decoding process and the generated frames are marked and inserted into the DPB as specified below.

Gaps in `frame_num` are detected by the decoding process and the generated frames are marked as specified in subclause 8.2.5.2.

After the marking of each generated frame, each picture m marked by the "sliding window" process as "unused for reference" is removed from the DPB when it is also marked as "non-existing" or its DPB output time is less than or equal to the CPB removal time of the current picture n ; i.e., $t_{o,dpb}(m) \leq t_r(n)$. When a frame or the last field in a frame buffer is removed from the DPB, the DPB fullness is decremented by one. The "non-existing" generated frame is inserted into the DPB and the DPB fullness is incremented by one.

C.2.2 Picture decoding and output

Picture n is decoded and its DPB output time $t_{o,dpb}(n)$ is derived by

$$t_{o,dpb}(n) = t_r(n) + t_c * \text{dpb_output_delay}(n) \quad (C-12)$$

The output of the current picture is specified as follows.

- If $t_{o,dpb}(n) = t_r(n)$, the current picture is output.

NOTE – When the current picture is a reference picture it will be stored in the DPB.

- Otherwise ($t_{o,dpb}(n) > t_r(n)$), the current picture is output later and will be stored in the DPB (as specified in subclause C.2.4) and is output at time $t_{o,dpb}(n)$ unless indicated not to be output by the decoding or inference of `no_output_of_prior_pics_flag` equal to 1 at a time that precedes $t_{o,dpb}(n)$.

The output picture shall be cropped, using the cropping rectangle specified in the sequence parameter set for the sequence.

When picture n is a picture that is output and is not the last picture of the bitstream that is output, the value of $\Delta t_{o,dpb}(n)$ is defined as:

$$\Delta t_{o,dpb}(n) = t_{o,dpb}(n_n) - t_{o,dpb}(n) \quad (C-13)$$

where n_n indicates the picture that follows after picture n in output order.

The decoded picture is temporarily stored (not in the DPB).

C.2.3 Removal of pictures from the DPB before possible insertion of the current picture

The removal of pictures from the DPB before possible insertion of the current picture proceeds as follows.

- If the decoded picture is an IDR picture the following applies.

- All reference pictures in the DPB are marked as "unused for reference" as specified in subclause 8.2.5.1.
- When the IDR picture is not the first IDR picture decoded and the value of `PicWidthInMbs` or `FrameHeightInMbs` or `max_dec_frame_buffering` derived from the active sequence parameter set is different from the value of `PicWidthInMbs` or `FrameHeightInMbs` or `max_dec_frame_buffering` derived from the sequence parameter set that was active for the preceding sequence, respectively, `no_output_of_prior_pics_flag` is inferred to be equal to 1 by the HRD, regardless of the actual value of `no_output_of_prior_pics_flag`.
NOTE – Decoder implementations should try to handle frame or DPB size changes more gracefully than the HRD in regard to changes in `PicWidthInMbs` or `FrameHeightInMbs`.
- When `no_output_of_prior_pics_flag` is equal to 1 or is inferred to be equal to 1, all frame buffers in the DPB are emptied without output of the pictures they contain, and DPB fullness is set to 0.
- Otherwise (the decoded picture is not an IDR picture), the following applies.
 - If the slice header of the current picture includes `memory_management_control_operation` equal to 5, all reference pictures in the DPB are marked as "unused for reference".
 - Otherwise (the slice header of the current picture does not include `memory_management_control_operation` equal to 5), the decoded reference picture marking process specified in subclause 8.2.5 is invoked.

All pictures m in the DPB, for which all of the following conditions are true, are removed from the DPB.

- picture m is marked as "unused for reference" or picture m is a non-reference picture. When a picture is a reference frame, it is considered to be marked as "unused for reference" only when both of its fields have been marked as "unused for reference".
- picture m is marked as "non-existing" or its DPB output time is less than or equal to the CPB removal time of the current picture n ; i.e., $t_{o,dpb}(m) \leq t_r(n)$

When a frame or the last field in a frame buffer is removed from the DPB, the DPB fullness is decremented by one.

C.2.4 Current decoded picture marking and storage

C.2.4.1 Marking and storage of a reference decoded picture into the DPB

When the current picture is a reference picture it is stored in the DPB as follows.

- If the current decoded picture is a second field (in decoding order) of a complementary reference field pair, and the first field of the pair is still in the DPB, the current decoded picture is stored in the same frame buffer as the first field of the pair.
- Otherwise, the current decoded picture is stored in an empty frame buffer, and the DPB fullness is incremented by one.

C.2.4.2 Storage of a non-reference picture into the DPB

When the current picture is a non-reference picture and current picture n has $t_{o,dpb}(n) > t_r(n)$, it is stored in the DPB as follows.

- If the current decoded picture is a second field (in decoding order) of a complementary non-reference field pair, and the first field of the pair is still in the DPB, the current decoded picture is stored in the same frame buffer as the first field of the pair.
- Otherwise, the current decoded picture is stored in an empty frame buffer, and the DPB fullness is incremented by one.

C.3 Bitstream conformance

A bitstream of coded data conforming to this Recommendation | International Standard fulfils the following requirements.

The bitstream is constructed according to the syntax, semantics, and constraints specified in this Recommendation | International Standard outside of this Annex.

The bitstream is tested by the HRD as specified below:

For Type I bitstreams, the number of tests carried out is equal to `cpb_cnt_minus1 + 1` where `cpb_cnt_minus1` is either the syntax element of `hrd_parameters()` following the `vcl_hrd_parameters_present_flag` or is determined by the application by other means not specified in this Recommendation | International Standard. One test is carried out for

each bit rate and CPB size combination specified by `hrd_parameters()` following the `vcl_hrd_parameters_present_flag`. Each of these tests is conducted at the Type I conformance point shown in Figure C-1.

For Type II bitstreams there are two sets of tests. The number of tests of the first set is equal to `cpb_cnt_minus1 + 1` where `cpb_cnt_minus1` is either the syntax element of `hrd_parameters()` following the `vcl_hrd_parameters_present_flag` or is determined by the application by other means not specified in this Recommendation | International Standard. One test is carried out for each bit rate and CPB size combination. Each of these tests is conducted at the Type I conformance point shown in Figure C-1. For these tests, only VCL and filler data NAL units are counted for the input bit rate and CPB storage.

The number of tests of the second set, for Type II bitstreams, is equal to `cpb_cnt_minus1 + 1` where `cpb_cnt_minus1` is either the syntax element of `hrd_parameters()` following the `nal_hrd_parameters_present_flag` or is determined by the application by other means not specified in this Recommendation | International Standard. One test is carried out for each bit rate and CPB size combination specified by `hrd_parameters()` following the `nal_hrd_parameters_present_flag`. Each of these tests is conducted at the Type II conformance point shown in Figure C-1. For these tests, all NAL units (of a Type II NAL unit stream) or all bytes (of a byte stream) are counted for the input bit rate and CPB storage.

NOTE 1 – NAL HRD parameters established by a value of `SchedSelIdx` for the Type II conformance point shown in Figure C-1 are sufficient to also establish VCL HRD conformance for the Type I conformance point shown in Figure C-1 for the same values of `initial_cpb_removal_delay[SchedSelIdx]`, `BitRate[SchedSelIdx]`, and `CpbSize[SchedSelIdx]` for the VBR case (`cbr_flag[SchedSelIdx]` equal to 0). This is because the data flow into the Type I conformance point is a subset of the data flow into the Type II conformance point and because, for the VBR case, the CPB is allowed to become empty and stay empty until the time a next picture is scheduled to begin to arrive. For example, when NAL HRD parameters are provided for the Type II conformance point that not only fall within the bounds set for NAL HRD parameters for profile conformance in item j of subclause A.3.1 or item i of subclause A.3.3 (depending on the profile in use) but also fall within the bounds set for VCL HRD parameters for profile conformance in item i of subclause A.3.1 or item h of subclause A.3.3 (depending on the profile in use), conformance of the VCL HRD for the Type I conformance point is also assured to fall within the bounds of item i of subclause A.3.1.

For conforming bitstreams, all of the following conditions shall be fulfilled for each of the tests.

- For each access unit n , with $n > 0$, associated with a buffering period SEI message, with $\Delta t_{g,90}(n)$ specified by

$$\Delta t_{g,90}(n) = 90000 * (t_{r,n}(n) - t_{af}(n-1)) \quad (C-14)$$

The value of `initial_cpb_removal_delay[SchedSelIdx]` shall be constrained as follows.

- If `cbr_flag[SchedSelIdx]` is equal to 0,

$$\text{initial_cpb_removal_delay[SchedSelIdx]} \leq \text{Ceil}(\Delta t_{g,90}(n)) \quad (C-15)$$

- Otherwise (`cbr_flag[SchedSelIdx]` is equal to 1),

$$\text{Floor}(\Delta t_{g,90}(n)) \leq \text{initial_cpb_removal_delay[SchedSelIdx]} \leq \text{Ceil}(\Delta t_{g,90}(n)) \quad (C-16)$$

NOTE 2 – The exact number of bits in the CPB at the removal time of each picture may depend on which buffering period SEI message is selected to initialize the HRD. Encoders must take this into account to ensure that all specified constraints must be obeyed regardless of which buffering period SEI message is selected to initialize the HRD, as the HRD may be initialised at any one of the buffering period SEI messages.

- A CPB overflow is specified as the condition in which the total number of bits in the CPB is larger than the CPB size. The CPB shall never overflow.
- A CPB underflow is specified as the condition in which $t_{r,n}(n)$ is less than $t_{af}(n)$. When `low_delay_hrd_flag` is equal to 0, the CPB shall never underflow.
- The nominal removal times of pictures from the CPB (starting from the second picture in decoding order), shall satisfy the constraints on $t_{r,n}(n)$ and $t_r(n)$ expressed in subclauses A.3.1 through A.3.3 for the profile and level specified in the bitstream.
- Immediately after any decoded picture is added to the DPB, the fullness of the DPB shall be less than or equal to the DPB size as constrained by Annexes A, D, and E for the profile and level specified in the bitstream.
- All reference pictures shall be present in the DPB when needed for prediction. Each picture shall be present in the DPB at its DPB output time unless it is not stored in the DPB at all, or is removed from the DPB before its output time by one of the processes specified in subclause C.2.

- The value of $\Delta_{to,dpb}(n)$ as given by Equation C-13, which is the difference between the output time of a picture and that of the picture immediately following it in output order, shall satisfy the constraint expressed in subclause A.3.1 for the profile and level specified in the bitstream.

C.4 Decoder conformance

A decoder conforming to this Recommendation | International Standard fulfils the following requirements.

A decoder claiming conformance to a specific profile and level shall be able to decode successfully all conforming bitstreams specified for decoder conformance in subclause C.3, provided that all sequence parameter sets and picture parameters sets referred to in the VCL NAL units, and appropriate buffering period and picture timing SEI messages are conveyed to the decoder, in a timely manner, either in the bitstream (by non-VCL NAL units), or by external means not specified by this Recommendation | International Standard.

There are two types of conformance that can be claimed by a decoder: output timing conformance and output order conformance.

To check conformance of a decoder, test bitstreams conforming to the claimed profile and level, as specified by subclause C.3 are delivered by a hypothetical stream scheduler (HSS) both to the HRD and to the decoder under test (DUT). All pictures output by the HRD shall also be output by the DUT and, for each picture output by the HRD, the values of all samples that are output by the DUT for the corresponding picture shall be equal to the values of the samples output by the HRD.

For output timing decoder conformance, the HSS operates as described above, with delivery schedules selected only from the subset of values of SchedSelIdx for which the bit rate and CPB size are restricted as specified in Annex A for the specified profile and level, or with "interpolated" delivery schedules as specified below for which the bit rate and CPB size are restricted as specified in Annex A. The same delivery schedule is used for both the HRD and DUT.

When the HRD parameters and the buffering period SEI messages are present with `cpb_cnt_minus1` greater than 0, the decoder shall be capable of decoding the bitstream as delivered from the HSS operating using an "interpolated" delivery schedule specified as having peak bit rate r , CPB size $c(r)$, and initial CPB removal delay $(f(r) \div r)$ as follows

$$\alpha = (r - \text{BitRate}[\text{SchedSelIdx} - 1]) \div (\text{BitRate}[\text{SchedSelIdx}] - \text{BitRate}[\text{SchedSelIdx} - 1]), \quad (\text{C-17})$$

$$c(r) = \alpha * \text{CpbSize}[\text{SchedSelIdx}] + (1 - \alpha) * \text{CpbSize}[\text{SchedSelIdx} - 1], \quad (\text{C-18})$$

$$f(r) = \alpha * \text{initial_cpb_removal_delay}[\text{SchedSelIdx}] * \text{BitRate}[\text{SchedSelIdx}] + (1 - \alpha) * \text{initial_cpb_removal_delay}[\text{SchedSelIdx} - 1] * \text{BitRate}[\text{SchedSelIdx} - 1] \quad (\text{C-19})$$

for any $\text{SchedSelIdx} > 0$ and r such that $\text{BitRate}[\text{SchedSelIdx} - 1] \leq r \leq \text{BitRate}[\text{SchedSelIdx}]$ such that r and $c(r)$ are within the limits as specified in Annex A for the maximum bit rate and buffer size for the specified profile and level.

NOTE 1 – `initial_cpb_removal_delay[SchedSelIdx]` can be different from one buffering period to another and have to be re-calculated.

For output timing decoder conformance, an HRD as described above is used and the timing (relative to the delivery time of the first bit) of picture output is the same for both HRD and the DUT up to a fixed delay.

For output order decoder conformance, the HSS delivers the bitstream to the DUT "by demand" from the DUT, meaning that the HSS delivers bits (in decoding order) only when the DUT requires more bits to proceed with its processing.

NOTE 2 – This means that for this test, the coded picture buffer of the DUT could be as small as the size of the largest access unit.

A modified HRD as described below is used, and the HSS delivers the bitstream to the HRD by one of the schedules specified in the bitstream such that the bit rate and CPB size are restricted as specified in Annex A. The order of pictures output shall be the same for both HRD and the DUT.

For output order decoder conformance, the HRD CPB size is equal to `CpbSize[SchedSelIdx]` for the selected schedule and the DPB size is equal to `MaxDpbSize`. Removal time from the CPB for the HRD is equal to final bit arrival time and decoding is immediate. The operation of the DPB of this HRD is described below.

C.4.1 Operation of the output order DPB

The decoded picture buffer contains frame buffers. Each of the frame buffers may contain a decoded frame, a decoded complementary field pair or a single (non-paired) decoded field that is marked as "used for reference" or is held for future output (reordered pictures). At HRD initialization, the DPB fullness, measured in frames, is set to 0. The following steps all happen instantaneously when an access unit is removed from the CPB, and in the order listed.

C.4.2 Decoding of gaps in frame_num and storage of "non-existing" pictures

When applicable, gaps in frame_num are detected by the decoding process and the necessary number of "non-existing" frames are inferred in the order specified by the generation of values of UnusedShortTermFrameNum in Equation 7-21 and are marked as specified in subclause 8.2.5.2. Frame buffers containing a frame or a complementary field pair or a non-paired field which are marked as "not needed for output" and "unused for reference" are emptied (without output), and the DPB fullness is decremented by the number of frame buffers emptied. Each "non-existing" frame is stored in the DPB as follows.

- When there is no empty frame buffer (i.e., DPB fullness is equal to DPB size), the "bumping" process specified in subclause C.4.5.3 is invoked repeatedly until there is an empty frame buffer in which to store the "non-existing" frame.
- The "non-existing" frame is stored in an empty frame buffer and is marked as "not needed for output", and the DPB fullness is incremented by one.

C.4.3 Picture decoding

Primary coded picture *n* is decoded and is temporarily stored (not in the DPB).

C.4.4 Removal of pictures from the DPB before possible insertion of the current picture

The removal of pictures from the DPB before possible insertion of the current picture proceeds as follows.

- If the decoded picture is an IDR picture the following applies.
 - All reference pictures in the DPB are marked as "unused for reference" as specified in subclause 8.2.5.
 - When the IDR picture is not the first IDR picture decoded and the value of PicWidthInMbs or FrameHeightInMbs or max_dec_frame_buffering derived from the active sequence parameter set is different from the value of PicWidthInMbs or FrameHeightInMbs or max_dec_frame_buffering derived from the sequence parameter set that was active for the preceding sequence, respectively, no_output_of_prior_pics_flag is inferred to be equal to 1 by the HRD, regardless of the actual value of no_output_of_prior_pics_flag.
NOTE – Decoder implementations should try to handle changes in the value of PicWidthInMbs or FrameHeightInMbs or max_dec_frame_buffering more gracefully than the HRD.
 - When no_output_of_prior_pics_flag is equal to 1 or is inferred to be equal to 1, all frame buffers in the DPB are emptied without output of the pictures they contain, and DPB fullness is set to 0.
- Otherwise (the decoded picture is not an IDR picture), the decoded reference picture marking process is invoked as specified in subclause 8.2.5. Frame buffers containing a frame or a complementary field pair or a non-paired field which are marked as "not needed for output" and "unused for reference" are emptied (without output), and the DPB fullness is decremented by the number of frame buffers emptied.

When the current picture has a memory_management_control_operation equal to 5 or is an IDR picture for which no_output_of_prior_pics_flag is not equal to 1 and is not inferred to be equal to 1, the following two steps are performed.

1. Frame buffers containing a frame or a complementary field pair or a non-paired field which are marked as "not needed for output" and "unused for reference" are emptied (without output), and the DPB fullness is decremented by the number of frame buffers emptied.
2. All non-empty frame buffers in the DPB are emptied by repeatedly invoking the "bumping" process specified in subclause C.4.5.3, and the DPB fullness is set to 0.

C.4.5 Current decoded picture marking and storage

C.4.5.1 Storage and marking of a reference decoded picture into the DPB

When the current picture is a reference picture, it is stored in the DPB as follows.

- If the current decoded picture is the second field (in decoding order) of a complementary reference field pair, and the first field of the pair is still in the DPB, the current picture is stored in the same frame buffer as the first field of the pair and is marked as "needed for output".

- Otherwise, the following operations are performed:
 - When there is no empty frame buffer (i.e., DPB fullness is equal to DPB size), the "bumping" process specified in subclause C.4.5.3 is invoked repeatedly until there is an empty frame buffer in which to store the current decoded picture.
 - The current decoded picture is stored in an empty frame buffer and is marked as "needed for output", and the DPB fullness is incremented by one.

C.4.5.2 Storage and marking of a non-reference decoded picture into the DPB

When the current picture is a non-reference picture, the following operations are performed.

- If the current decoded picture is the second field (in decoding order) of a complementary non-reference field pair and the first field of the pair is still in the DPB, the current picture is stored in the same frame buffer as the first field of the pair and is marked as "needed for output".
- Otherwise, the following operations are performed repeatedly until the current decoded picture has been cropped and output or has been stored in the DPB:
 - If there is no empty frame buffer (i.e., DPB fullness is equal to DPB size), the following applies.
 - If the current picture does not have a lower value of `PicOrderCnt()` than all pictures in the DPB that are marked as "needed for output", the "bumping" process described in subclause C.4.5.3 is performed.
 - Otherwise (the current picture has a lower value of `PicOrderCnt()` than all pictures in the DPB that are marked as "needed for output"), the current picture is cropped, using the cropping rectangle specified in the sequence parameter set for the sequence and the cropped picture is output.
 - Otherwise (there is an empty frame buffer, i.e., DPB fullness is less than DPB size) the current decoded picture is stored in an empty frame buffer and is marked as "needed for output", and the DPB fullness is incremented by one.

C.4.5.3 "Bumping" process

The "bumping" process is invoked in the following cases.

- There is no empty frame buffer (i.e., DPB fullness is equal to DPB size) and an empty frame buffer is needed for storage of an inferred "non-existing" frame, as specified in subclause C.4.2.
- The current picture is an IDR picture and `no_output_of_prior_pics_flag` is not equal to 1 and is not inferred to be equal to 1, as specified in subclause C.4.4.
- The current picture has `memory_management_control_operation` equal to 5, as specified in subclause C.4.4.
- There is no empty frame buffer (i.e., DPB fullness is equal to DPB size) and an empty frame buffer is needed for storage of a decoded (non-IDR) reference picture, as specified in subclause C.4.5.1.
- There is no empty frame buffer (i.e., DPB fullness is equal to DPB size) and the current picture is a non-reference picture that is not the second field of a complementary non-reference field pair and there are pictures in the DPB that are marked as "needed for output" that precede the current non-reference picture in output order, as specified in subclause C.4.5.2, so an empty buffer is needed for storage of the current picture.

The "bumping" process consists of the following:

- The picture or complementary reference field pair that is first for output is selected as follows.
 - The frame buffer is selected that contains the picture having the smallest value of `PicOrderCnt()` of all pictures in the DPB marked as "needed for output".
 - If this frame buffer contains a complementary non-reference field pair with both fields marked as "needed for output" and both fields have the same `PicOrderCnt()`, the first of these two fields in decoding order is considered first for output.
 - Otherwise, if this frame buffer contains a complementary reference field pair with both fields marked as "needed for output" and both fields have the same `PicOrderCnt()`, the entire complementary reference field pair is considered first for output.
 - Otherwise, the picture in this frame buffer that has the smallest value of `PicOrderCnt()` is considered first for output.
- If a single picture is considered first for output, this picture is cropped, using the cropping rectangle specified in the sequence parameter set for the sequence, the cropped picture is output, and the picture is marked as "not needed for output".

- Otherwise (a complementary reference field pair is considered first for output), the two fields of the complementary reference field pair are both cropped, using the cropping rectangle specified in the sequence parameter set for the sequence, the two fields of the complementary reference field pair are output together, and both fields of the complementary reference field pair are marked as "not needed for output".
- The frame buffer that included the picture or complementary reference field pair that was cropped and output is checked, and when any of the following conditions is satisfied, the frame buffer is emptied and the DPB fullness is decremented by 1.
 - The frame buffer contains a non-reference non-paired field.
 - The frame buffer contains a non-reference frame.
 - The frame buffer contains a complementary non-reference field pair with both fields marked as "not needed for output".
 - The frame buffer contains a non-paired reference field marked as "unused for reference".
 - The frame buffer contains a reference frame with both fields marked as "unused for reference".
 - The frame buffer contains a complementary reference field pair with both fields marked as "unused for reference" and "not needed for output".

Annex D

Supplemental enhancement information

(This annex forms an integral part of this Recommendation | International Standard)

This annex specifies syntax and semantics for SEI message payloads.

SEI messages assist in processes related to decoding, display or other purposes. However, SEI messages are not required for constructing the luma or chroma samples by the decoding process. Conforming decoders are not required to process this information for output order conformance to this Recommendation | International Standard (see Annex C for the specification of conformance). Some SEI message information is required to check bitstream conformance and for output timing decoder conformance.

In Annex D, specification for presence of SEI messages are also satisfied when those messages (or some subset of them) are conveyed to decoders (or to the HRD) by other means not specified by this Recommendation | International Standard. When present in the bitstream, SEI messages shall obey the syntax and semantics specified in subclauses 7.3.2.3 and 7.4.2.3 and this annex. When the content of an SEI message is conveyed for the application by some means other than presence within the bitstream, the representation of the content of the SEI message is not required to use the same syntax specified in this annex. For the purpose of counting bits, only the appropriate bits that are actually present in the bitstream are counted.

D.1 SEI payload syntax

sei_payload(payloadType, payloadSize) {	C	Descriptor
if(payloadType == 0)		
buffering_period(payloadSize)	5	
else if(payloadType == 1)		
pic_timing(payloadSize)	5	
else if(payloadType == 2)		
pan_scan_rect(payloadSize)	5	
else if(payloadType == 3)		
filler_payload(payloadSize)	5	
else if(payloadType == 4)		
user_data_registered_itu_t_t35(payloadSize)	5	
else if(payloadType == 5)		
user_data_unregistered(payloadSize)	5	
else if(payloadType == 6)		
recovery_point(payloadSize)	5	
else if(payloadType == 7)		
dec_ref_pic_marking_repetition(payloadSize)	5	
else if(payloadType == 8)		
spare_pic(payloadSize)	5	
else if(payloadType == 9)		
scene_info(payloadSize)	5	
else if(payloadType == 10)		
sub_seq_info(payloadSize)	5	
else if(payloadType == 11)		
sub_seq_layer_characteristics(payloadSize)	5	
else if(payloadType == 12)		
sub_seq_characteristics(payloadSize)	5	
else if(payloadType == 13)		
full_frame_freeze(payloadSize)	5	
else if(payloadType == 14)		
full_frame_freeze_release(payloadSize)	5	
else if(payloadType == 15)		
full_frame_snapshot(payloadSize)	5	
else if(payloadType == 16)		
progressive_refinement_segment_start(payloadSize)	5	
else if(payloadType == 17)		
progressive_refinement_segment_end(payloadSize)	5	
else if(payloadType == 18)		
motion_constrained_slice_group_set(payloadSize)	5	
else if(payloadType == 19)		
film_grain_characteristics(payloadSize)	5	
else if(payloadType == 20)		
deblocking_filter_display_preference(payloadSize)	5	
else if(payloadType == 21)		
stereo_video_info(payloadSize)	5	
else		
reserved_sei_message(payloadSize)	5	

if(!byte_aligned()) {		
bit_equal_to_one /* equal to 1 */	5	f(1)
while(!byte_aligned())		
bit_equal_to_zero /* equal to 0 */	5	f(1)
}		
}		

D.1.1 Buffering period SEI message syntax

buffering_period(payloadSize) {	C	Descriptor
seq_parameter_set_id	5	ue(v)
if(NalHrdBpPresentFlag) {		
for(SchedSelIdx = 0; SchedSelIdx <= cpb_cnt_minus1; SchedSelIdx++) {		
initial_cpb_removal_delay [SchedSelIdx]	5	u(v)
initial_cpb_removal_delay_offset [SchedSelIdx]	5	u(v)
}		
}		
if(VclHrdBpPresentFlag) {		
for(SchedSelIdx = 0; SchedSelIdx <= cpb_cnt_minus1; SchedSelIdx++) {		
initial_cpb_removal_delay [SchedSelIdx]	5	u(v)
initial_cpb_removal_delay_offset [SchedSelIdx]	5	u(v)
}		
}		
}		

D.1.2 Picture timing SEI message syntax

pic_timing(payloadSize) {	C	Descriptor
if(CpbDpbDelaysPresentFlag) {		
cpb_removal_delay	5	u(v)
dpb_output_delay	5	u(v)
}		
if(pic_struct_present_flag) {		
pic_struct	5	u(4)
for(i = 0; i < NumClockTS ; i++) {		
clock_timestamp_flag [i]	5	u(1)
if(clock_timestamp_flag[i]) {		
ct_type	5	u(2)
nuit_field_based_flag	5	u(1)
counting_type	5	u(5)
full_timestamp_flag	5	u(1)
discontinuity_flag	5	u(1)
cnt_dropped_flag	5	u(1)
n_frames	5	u(8)
if(full_timestamp_flag) {		
seconds_value /* 0..59 */	5	u(6)
}		

minutes_value /* 0..59 */	5	u(6)
hours_value /* 0..23 */	5	u(5)
} else {		
seconds_flag	5	u(1)
if(seconds_flag) {		
seconds_value /* range 0..59 */	5	u(6)
minutes_flag	5	u(1)
if(minutes_flag) {		
minutes_value /* 0..59 */	5	u(6)
hours_flag	5	u(1)
if(hours_flag)		
hours_value /* 0..23 */	5	u(5)
}		
}		
}		
if(time_offset_length > 0)		
time_offset	5	i(v)
}		
}		
}		
}		

D.1.3 Pan-scan rectangle SEI message syntax

pan_scan_rect(payloadSize) {	C	Descriptor
pan_scan_rect_id	5	ue(v)
pan_scan_rect_cancel_flag	5	u(1)
if(!pan_scan_rect_cancel_flag) {		
pan_scan_cnt_minus1	5	ue(v)
for(i = 0; i <= pan_scan_cnt_minus1; i++) {		
pan_scan_rect_left_offset[i]	5	se(v)
pan_scan_rect_right_offset[i]	5	se(v)
pan_scan_rect_top_offset[i]	5	se(v)
pan_scan_rect_bottom_offset[i]	5	se(v)
}		
pan_scan_rect_repetition_period	5	ue(v)
}		
}		

D.1.4 Filler payload SEI message syntax

filler_payload(payloadSize) {	C	Descriptor
for(k = 0; k < payloadSize; k++)		
ff_byte /* equal to 0xFF */	5	f(8)
}		

D.1.5 User data registered by ITU-T Rec. T.35 SEI message syntax

user_data_registered_itu_t_t35(payloadSize) {	C	Descriptor
itu_t_t35_country_code	5	b(8)
if(itu_t_t35_country_code != 0xFF)		
i = 1		
else {		
itu_t_t35_country_code_extension_byte	5	b(8)
i = 2		
}		
do {		
itu_t_t35_payload_byte	5	b(8)
i++		
} while(i < payloadSize)		
}		

D.1.6 User data unregistered SEI message syntax

user_data_unregistered(payloadSize) {	C	Descriptor
uuid_iso_iec_11578	5	u(128)
for(i = 16; i < payloadSize; i++)		
user_data_payload_byte	5	b(8)
}		

D.1.7 Recovery point SEI message syntax

recovery_point(payloadSize) {	C	Descriptor
recovery_frame_cnt	5	ue(v)
exact_match_flag	5	u(1)
broken_link_flag	5	u(1)
changing_slice_group_idc	5	u(2)
}		

D.1.8 Decoded reference picture marking repetition SEI message syntax

dec_ref_pic_marking_repetition(payloadSize) {	C	Descriptor
original_idr_flag	5	u(1)
original_frame_num	5	ue(v)
if(!frame_mbs_only_flag) {		
original_field_pic_flag	5	u(1)
if(original_field_pic_flag)		
original_bottom_field_flag	5	u(1)
}		
dec_ref_pic_marking()	5	
}		

D.1.9 Spare picture SEI message syntax

spare_pic(payloadSize) {	C	Descriptor
target_frame_num	5	ue(v)
spare_field_flag	5	u(1)
if(spare_field_flag)		
target_bottom_field_flag	5	u(1)
num_spare_pics_minus1	5	ue(v)
for(i = 0; i < num_spare_pics_minus1 + 1; i++) {		
delta_spare_frame_num[i]	5	ue(v)
if(spare_field_flag)		
spare_bottom_field_flag[i]	5	u(1)
spare_area_idc[i]	5	ue(v)
if(spare_area_idc[i] == 1)		
for(j = 0; j < PicSizeInMapUnits; j++)		
spare_unit_flag[i][j]	5	u(1)
else if(spare_area_idc[i] == 2) {		
mapUnitCnt = 0		
for(j=0; mapUnitCnt < PicSizeInMapUnits; j++) {		
zero_run_length[i][j]	5	ue(v)
mapUnitCnt += zero_run_length[i][j] + 1		
}		
}		
}		
}		

D.1.10 Scene information SEI message syntax

scene_info(payloadSize) {	C	Descriptor
scene_info_present_flag	5	u(1)
if(scene_info_present_flag) {		
scene_id	5	ue(v)
scene_transition_type	5	ue(v)
if(scene_transition_type > 3)		
second_scene_id	5	ue(v)
}		
}		

D.1.11 Sub-sequence information SEI message syntax

sub_seq_info(payloadSize) {	C	Descriptor
sub_seq_layer_num	5	ue(v)
sub_seq_id	5	ue(v)
first_ref_pic_flag	5	u(1)
leading_non_ref_pic_flag	5	u(1)
last_pic_flag	5	u(1)
sub_seq_frame_num_flag	5	u(1)
if(sub_seq_frame_num_flag)		
sub_seq_frame_num	5	ue(v)
}		

D.1.12 Sub-sequence layer characteristics SEI message syntax

sub_seq_layer_characteristics(payloadSize) {	C	Descriptor
num_sub_seq_layers_minus1	5	ue(v)
for(layer = 0; layer <= num_sub_seq_layers_minus1; layer++) {		
accurate_statistics_flag	5	u(1)
average_bit_rate	5	u(16)
average_frame_rate	5	u(16)
}		
}		

D.1.13 Sub-sequence characteristics SEI message syntax

sub_seq_characteristics(payloadSize) {	C	Descriptor
sub_seq_layer_num	5	ue(v)
sub_seq_id	5	ue(v)
duration_flag	5	u(1)
if(duration_flag)		
sub_seq_duration	5	u(32)
average_rate_flag	5	u(1)
if(average_rate_flag) {		
accurate_statistics_flag	5	u(1)
average_bit_rate	5	u(16)
average_frame_rate	5	u(16)
}		
num_referenced_subseqs	5	ue(v)
for(n = 0; n < num_referenced_subseqs; n++) {		
ref_sub_seq_layer_num	5	ue(v)
ref_sub_seq_id	5	ue(v)
ref_sub_seq_direction	5	u(1)
}		
}		

D.1.14 Full-frame freeze SEI message syntax

full_frame_freeze(payloadSize) {	C	Descriptor
full_frame_freeze_repetition_period	5	ue(v)
}		

D.1.15 Full-frame freeze release SEI message syntax

full_frame_freeze_release(payloadSize) {	C	Descriptor
}		

D.1.16 Full-frame snapshot SEI message syntax

full_frame_snapshot(payloadSize) {	C	Descriptor
snapshot_id	5	ue(v)
}		

D.1.17 Progressive refinement segment start SEI message syntax

progressive_refinement_segment_start(payloadSize) {	C	Descriptor
progressive_refinement_id	5	ue(v)
num_refinement_steps_minus1	5	ue(v)
}		

D.1.18 Progressive refinement segment end SEI message syntax

progressive_refinement_segment_end(payloadSize) {	C	Descriptor
progressive_refinement_id	5	ue(v)
}		

D.1.19 Motion-constrained slice group set SEI message syntax

motion_constrained_slice_group_set(payloadSize) {	C	Descriptor
num_slice_groups_in_set_minus1	5	ue(v)
for(i = 0; i <= num_slice_groups_in_set_minus1; i++)		
slice_group_id[i]	5	u(v)
exact_sample_value_match_flag	5	u(1)
pan_scan_rect_flag	5	u(1)
if(pan_scan_rect_flag)		
pan_scan_rect_id	5	ue(v)
}		

D.1.20 Film grain characteristics SEI message syntax

film_grain_characteristics(payloadSize) {	C	Descriptor
film_grain_characteristics_cancel_flag	5	u(1)
if(!film_grain_characteristics_cancel_flag) {		
model_id	5	u(2)
separate_colour_description_present_flag	5	u(1)
if(separate_colour_description_present_flag) {		
film_grain_bit_depth_luma_minus8	5	u(3)
film_grain_bit_depth_chroma_minus8	5	u(3)
film_grain_full_range_flag	5	u(1)
film_grain_colour_primaries	5	u(8)
film_grain_transfer_characteristics	5	u(8)
film_grain_matrix_coefficients	5	u(8)
}		
blending_mode_id	5	u(2)
log2_scale_factor	5	u(4)
for(c = 0; c < 3; c++)		
comp_model_present_flag[c]	5	u(1)
for(c = 0; c < 3; c++)		
if(comp_model_present_flag[c]) {		
num_intensity_intervals_minus1[c]	5	u(8)
num_model_values_minus1[c]	5	u(3)
for(i = 0; i <= num_intensity_intervals_minus1[c]; i++) {		
intensity_interval_lower_bound[c][i]	5	u(8)
intensity_interval_upper_bound[c][i]	5	u(8)
for(j = 0; j <= num_model_values_minus1[c]; j++)		
comp_model_value[c][i][j]	5	se(v)
}		
}		
}		
film_grain_characteristics_repetition_period	5	ue(v)
}		
}		

D.1.21 Deblocking filter display preference SEI message syntax

deblocking_filter_display_preference(payloadSize) {	C	Descriptor
deblocking_display_preference_cancel_flag	5	u(1)
if(!deblocking_display_preference_cancel_flag) {		
display_prior_to_deblocking_preferred_flag	5	u(1)
dec_frame_buffering_constraint_flag	5	u(1)
deblocking_display_preference_repetition_period	5	ue(v)
}		
}		

D.1.22 Stereo video information SEI message syntax

stereo_video_info(payloadSize) {	C	Descriptor
field_views_flag	5	u(1)
if(field_views_flag)		
top_field_is_left_view_flag	5	u(1)
else {		
current_frame_is_left_view_flag	5	u(1)
next_frame_is_second_view_flag	5	u(1)
}		
left_view_self_contained_flag	5	u(1)
right_view_self_contained_flag	5	u(1)
}		

D.1.23 Reserved SEI message syntax

reserved_sei_message(payloadSize) {	C	Descriptor
for(i = 0; i < payloadSize; i++)		
reserved_sei_message_payload_byte	5	b(8)
}		

D.2 SEI payload semantics

D.2.1 Buffering period SEI message semantics

When NalHrdBpPresentFlag or VclHrdBpPresentFlag are equal to 1, a buffering period SEI message can be associated with any access unit in the bitstream, and a buffering period SEI message shall be associated with each IDR access unit and with each access unit associated with a recovery point SEI message.

NOTE – For some applications, the frequent presence of a buffering period SEI message may be desirable.

A buffering period is specified as the set of access units between two instances of the buffering period SEI message in decoding order.

seq_parameter_set_id specifies the sequence parameter set that contains the sequence HRD attributes. The value of seq_parameter_set_id shall be equal to the value of seq_parameter_set_id in the picture parameter set referenced by the primary coded picture associated with the buffering period SEI message. The value of seq_parameter_set_id shall be in the range of 0 to 31, inclusive.

initial_cpb_removal_delay[SchedSelIdx] specifies the delay for the SchedSelIdx-th CPB between the time of arrival in the CPB of the first bit of the coded data associated with the access unit associated with the buffering period SEI message and the time of removal from the CPB of the coded data associated with the same access unit, for the first buffering period after HRD initialisation. The syntax element has a length in bits given by initial_cpb_removal_delay_length_minus1 + 1. It is in units of a 90 kHz clock. initial_cpb_removal_delay[SchedSelIdx] shall not be equal to 0 and shall not exceed $90000 * (CpbSize[SchedSelIdx] \div BitRate[SchedSelIdx])$, the time-equivalent of the CPB size in 90 kHz clock units.

initial_cpb_removal_delay_offset[SchedSelIdx] is used for the SchedSelIdx-th CPB in combination with the cpb_removal_delay to specify the initial delivery time of coded access units to the CPB. initial_cpb_removal_delay_offset[SchedSelIdx] is in units of a 90 kHz clock. The initial_cpb_removal_delay_offset[SchedSelIdx] syntax element is a fixed length code whose length in bits is given by initial_cpb_removal_delay_length_minus1 + 1. This syntax element is not used by decoders and is needed only for the delivery scheduler (HSS) specified in Annex C.

Over the entire coded video sequence, the sum of initial_cpb_removal_delay[SchedSelIdx] and initial_cpb_removal_delay_offset[SchedSelIdx] shall be constant for each value of SchedSelIdx.

D.2.2 Picture timing SEI message semantics

The presence of picture timing SEI message in the bitstream is specified as follows.

- If CpbDpbDelaysPresentFlag is equal to 1 or pic_struct_present_flag is equal to 1, one picture timing SEI message shall be present in every access unit of the coded video sequence.
- Otherwise (CpbDpbDelaysPresentFlag is equal to 0 and pic_struct_present_flag is equal to 0), no picture timing SEI messages shall be present in any access unit of the coded video sequence.

cpb_removal_delay specifies how many clock ticks (see subclause E.2.1) to wait after removal from the CPB of the access unit associated with the most recent buffering period SEI message before removing from the buffer the access unit data associated with the picture timing SEI message. This value is also used to calculate an earliest possible time of arrival of access unit data into the CPB for the HSS, as specified in Annex C. The syntax element is a fixed length code whose length in bits is given by $\text{cpb_removal_delay_length_minus1} + 1$. The **cpb_removal_delay** is the remainder of a $2^{(\text{cpb_removal_delay_length_minus1} + 1)}$ counter.

The value of **cpb_removal_delay** for the first picture in the bitstream shall be equal to 0.

dpb_output_delay is used to compute the DPB output time of the picture. It specifies how many clock ticks to wait after removal of an access unit from the CPB before the decoded picture can be output from the DPB (see subclause C.2).

NOTE 1 – A picture is not removed from the DPB at its output time when it is still marked as "used for short-term reference" or "used for long-term reference".

NOTE 2 – Only one **dpb_output_delay** is specified for a decoded picture.

The size of the syntax element **dpb_output_delay** is given in bits by $\text{dpb_output_delay_length_minus1} + 1$. When **max_dec_frame_buffering** is equal to 0, **dpb_output_delay** shall be equal to 0.

The output time derived from the **dpb_output_delay** of any picture that is output from an output timing conforming decoder as specified in subclause C.2 shall precede the output time derived from the **dpb_output_delay** of all pictures in any subsequent coded video sequence in decoding order.

The output time derived from the **dpb_output_delay** of the second field, in decoding order, of a complementary non-reference field pair shall exceed the output time derived from the **dpb_output_delay** of the first field of the same complementary non-reference field pair.

The picture output order established by the values of this syntax element shall be the same order as established by the values of **PicOrderCnt()** as specified by subclauses C.4.1 to C.4.5, except that when the two fields of a complementary reference field pair have the same value of **PicOrderCnt()**, the two fields have different output times.

For pictures that are not output by the "bumping" process of subclause C.4.5 because they precede, in decoding order, an IDR picture with **no_output_of_prior_pics_flag** equal to 1 or inferred to be equal to 1, the output times derived from **dpb_output_delay** shall be increasing with increasing value of **PicOrderCnt()** relative to all pictures within the same coded video sequence subsequent to any picture having a **memory_management_control_operation** equal to 5.

pic_struct indicates whether a picture should be displayed as a frame or one or more fields, according to Table D-1. Frame doubling (**pic_struct** equal to 7) indicates that the frame should be displayed two times consecutively, and frame tripling (**pic_struct** equal to 8) indicates that the frame should be displayed three times consecutively.

NOTE 3 – Frame doubling can facilitate the display, for example, of 25p video on a 50p display and 29.97p video on a 59.94p display. Using frame doubling and frame tripling in combination on every other frame can facilitate the display of 23.98p video on a 59.94p display.

Table D-1 – Interpretation of pic_struct

Value	Indicated display of picture	Restrictions	NumClockTS
0	frame	field_pic_flag shall be 0	1
1	top field	field_pic_flag shall be 1, bottom_field_flag shall be 0	1
2	bottom field	field_pic_flag shall be 1, bottom_field_flag shall be 1	1
3	top field, bottom field, in that order	field_pic_flag shall be 0	2
4	bottom field, top field, in that order	field_pic_flag shall be 0	2
5	top field, bottom field, top field repeated, in that order	field_pic_flag shall be 0	3
6	bottom field, top field, bottom field repeated, in that order	field_pic_flag shall be 0	3
7	frame doubling	field_pic_flag shall be 0 fixed_frame_rate_flag shall be 1	2
8	frame tripling	field_pic_flag shall be 0 fixed_frame_rate_flag shall be 1	3
9..15	reserved		

NumClockTS is determined by pic_struct as specified in Table D-1. There are up to NumClockTS sets of clock timestamp information for a picture, as specified by clock_timestamp_flag[i] for each set. The sets of clock timestamp information apply to the field(s) or the frame(s) associated with the picture by pic_struct.

The contents of the clock timestamp syntax elements indicate a time of origin, capture, or alternative ideal display. This indicated time is computed as

$$\text{clockTimestamp} = ((\text{hH} * 60 + \text{mM}) * 60 + \text{sS}) * \text{time_scale} + \text{nFrames} * (\text{num_units_in_tick} * (1 + \text{nuit_field_based_flag})) + \text{tOffset}, \quad (\text{D-1})$$

in units of clock ticks of a clock with clock frequency equal to time_scale Hz, relative to some unspecified point in time for which clockTimestamp is equal to 0. Output order and DPB output timing are not affected by the value of clockTimestamp. When two or more frames with pic_struct equal to 0 are consecutive in output order and have equal values of clockTimestamp, the indication is that the frames represent the same content and that the last such frame in output order is the preferred representation.

NOTE 4 – clockTimestamp time indications may aid display on devices with refresh rates other than those well-matched to DPB output times.

clock_timestamp_flag[i] equal to 1 indicates that a number of clock timestamp syntax elements are present and follow immediately. clock_timestamp_flag[i] equal to 0 indicates that the associated clock timestamp syntax elements are not present. When NumClockTS is greater than 1 and clock_timestamp_flag[i] is equal to 1 for more than one value of i, the value of clockTimestamp shall be non-decreasing with increasing value of i.

ct_type indicates the scan type (interlaced or progressive) of the source material as specified in Table D-2.

Two fields of a coded frame may have different values of ct_type.

When clockTimestamp is equal for two fields of opposite parity that are consecutive in output order, both with ct_type equal to 0 (progressive) or ct_type equal to 2 (unknown), the two fields are indicated to have come from the same original progressive frame. Two consecutive fields in output order shall have different values of clockTimestamp when the value of ct_type for either field is 1 (interlaced).

Table D-2 – Mapping of ct_type to source picture scan

Value	Original picture scan
0	progressive
1	interlaced
2	unknown
3	reserved

nuit_field_based_flag is used in calculating clockTimestamp, as specified in Equation D-1.

counting_type specifies the method of dropping values of the n_frames as specified in Table D-3.

Table D-3 – Definition of counting_type values

Value	Interpretation
0	no dropping of n_frames count values and no use of time_offset
1	no dropping of n_frames count values
2	dropping of individual zero values of n_frames count
3	dropping of individual MaxFPS-1 values of n_frames count
4	dropping of the two lowest (value 0 and 1) n_frames counts when seconds_value is equal to 0 and minutes_value is not an integer multiple of 10
5	dropping of unspecified individual n_frames count values
6	dropping of unspecified numbers of unspecified n_frames count values
7..31	reserved

full_timestamp_flag equal to 1 specifies that the n_frames syntax element is followed by seconds_value, minutes_value, and hours_value. full_timestamp_flag equal to 0 specifies that the n_frames syntax element is followed by seconds_flag.

discontinuity_flag equal to 0 indicates that the difference between the current value of clockTimestamp and the value of clockTimestamp computed from the previous clock timestamp in output order can be interpreted as the time difference between the times of origin or capture of the associated frames or fields. discontinuity_flag equal to 1 indicates that the difference between the current value of clockTimestamp and the value of clockTimestamp computed from the previous clock timestamp in output order should not be interpreted as the time difference between the times of origin or capture of the associated frames or fields. When discontinuity_flag is equal to 0, the value of clockTimestamp shall be greater than or equal to all values of clockTimestamp present for the preceding picture in DPB output order.

cnt_dropped_flag specifies the skipping of one or more values of n_frames using the counting method specified by counting_type.

n_frames specifies the value of nFrames used to compute clockTimestamp. n_frames shall be less than

$$\text{MaxFPS} = \text{Ceil}(\text{time_scale} \div \text{num_units_in_tick}) \quad (\text{D-2})$$

NOTE 5 – n_frames is a frame-based counter. For field-specific timing indications, time_offset should be used to indicate a distinct clockTimestamp for each field.

When `counting_type` is equal to 2 and `cnt_dropped_flag` is equal to 1, `n_frames` shall be equal to 1 and the value of `n_frames` for the previous picture in output order shall not be equal to 0 unless `discontinuity_flag` is equal to 1.

NOTE 6 – When `counting_type` is equal to 2, the need for increasingly large magnitudes of `tOffset` in Equation D-1 when using fixed non-integer frame rates (e.g., 12.5 frames per second with `time_scale` equal to 25 and `num_units_in_tick` equal to 2 and `nuit_field_based_flag` equal to 0) can be avoided by occasionally skipping over the value `n_frames` equal to 0 when counting (e.g., counting `n_frames` from 0 to 12, then incrementing `seconds_value` and counting `n_frames` from 1 to 12, then incrementing `seconds_value` and counting `n_frames` from 0 to 12, etc.).

When `counting_type` is equal to 3 and `cnt_dropped_flag` is equal to 1, `n_frames` shall be equal to 0 and the value of `n_frames` for the previous picture in output order shall not be equal to `MaxFPS` – 1 unless `discontinuity_flag` is equal to 1.

NOTE 7 – When `counting_type` is equal to 3, the need for increasingly large magnitudes of `tOffset` in Equation D-1 when using fixed non-integer frame rates (e.g., 12.5 frames per second with `time_scale` equal to 25 and `num_units_in_tick` equal to 2 and `nuit_field_based_flag` equal to 0) can be avoided by occasionally skipping over the value `n_frames` equal to `MaxFPS` when counting (e.g., counting `n_frames` from 0 to 12, then incrementing `seconds_value` and counting `n_frames` from 0 to 11, then incrementing `seconds_value` and counting `n_frames` from 0 to 12, etc.).

When `counting_type` is equal to 4 and `cnt_dropped_flag` is equal to 1, `n_frames` shall be equal to 2 and the specified value of `sS` shall be zero and the specified value of `mM` shall not be an integer multiple of ten and `n_frames` for the previous picture in output order shall not be equal to 0 or 1 unless `discontinuity_flag` is equal to 1.

NOTE 8 – When `counting_type` is equal to 4, the need for increasingly large magnitudes of `tOffset` in Equation D-1 when using fixed non-integer frame rates (e.g., 30000÷1001 frames per second with `time_scale` equal to 60000 and `num_units_in_tick` equal to 1 001 and `nuit_field_based_flag` equal to 1) can be reduced by occasionally skipping over the value `n_frames` equal to `MaxFPS` when counting (e.g., counting `n_frames` from 0 to 29, then incrementing `seconds_value` and counting `n_frames` from 0 to 29, etc., until the `seconds_value` is zero and `minutes_value` is not an integer multiple of ten, then counting `n_frames` from 2 to 29, then incrementing `seconds_value` and counting `n_frames` from 0 to 29, etc.). This counting method is well known in industry and is often referred to as "NTSC drop-frame" counting.

When `counting_type` is equal to 5 or 6 and `cnt_dropped_flag` is equal to 1, `n_frames` shall not be equal to 1 plus the value of `n_frames` for the previous picture in output order modulo `MaxFPS` unless `discontinuity_flag` is equal to 1.

NOTE 9 – When `counting_type` is equal to 5 or 6, the need for increasingly large magnitudes of `tOffset` in Equation D-1 when using fixed non-integer frame rates can be avoided by occasionally skipping over some values of `n_frames` when counting. The specific values of `n_frames` that are skipped are not specified when `counting_type` is equal to 5 or 6.

seconds_flag equal to 1 specifies that `seconds_value` and `minutes_flag` are present when `full_timestamp_flag` is equal to 0. `seconds_flag` equal to 0 specifies that `seconds_value` and `minutes_flag` are not present.

seconds_value specifies the value of `sS` used to compute `clockTimestamp`. The value of `seconds_value` shall be in the range of 0 to 59, inclusive. When `seconds_value` is not present, the previous `seconds_value` in decoding order shall be used as `sS` to compute `clockTimestamp`.

minutes_flag equal to 1 specifies that `minutes_value` and `hours_flag` are present when `full_timestamp_flag` is equal to 0 and `seconds_flag` is equal to 1. `minutes_flag` equal to 0 specifies that `minutes_value` and `hours_flag` are not present.

minutes_value specifies the value of `mM` used to compute `clockTimestamp`. The value of `minutes_value` shall be in the range of 0 to 59, inclusive. When `minutes_value` is not present, the previous `minutes_value` in decoding order shall be used as `mM` to compute `clockTimestamp`.

hours_flag equal to 1 specifies that `hours_value` is present when `full_timestamp_flag` is equal to 0 and `seconds_flag` is equal to 1 and `minutes_flag` is equal to 1.

hours_value specifies the value of `hH` used to compute `clockTimestamp`. The value of `hours_value` shall be in the range of 0 to 23, inclusive. When `hours_value` is not present, the previous `hours_value` in decoding order shall be used as `hH` to compute `clockTimestamp`.

time_offset specifies the value of `tOffset` used to compute `clockTimestamp`. The number of bits used to represent `time_offset` shall be equal to `time_offset_length`. When `time_offset` is not present, the value 0 shall be used as `tOffset` to compute `clockTimestamp`.

D.2.3 Pan-scan rectangle SEI message semantics

The pan-scan rectangle SEI message syntax elements specify the coordinates of a rectangle relative to the cropping rectangle of the sequence parameter set. Each coordinate of this rectangle is specified in units of one-sixteenth sample spacing relative to the luma sampling grid.

pan_scan_rect_id contains an identifying number that may be used to identify the purpose of the pan-scan rectangle (for example, to identify the rectangle as the area to be shown on a particular display device or as the area that contains a particular actor in the scene). The value of `pan_scan_rect_id` shall be in the range of 0 to $2^{32} - 1$, inclusive.

Values of `pan_scan_rect_id` from 0 to 255 and from 512 to $2^{31}-1$ may be used as determined by the application. Values of `pan_scan_rect_id` from 256 to 511 and from 2^{31} to $2^{32}-1$ are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of `pan_scan_rect_id` in the range of 256 to 511 or in the range of 2^{31} to $2^{32}-1$ shall ignore (remove from the bitstream and discard) it.

`pan_scan_rect_cancel_flag` equal to 1 indicates that the SEI message cancels the persistence of any previous pan-scan rectangle SEI message in output order. `pan_scan_rect_cancel_flag` equal to 0 indicates that pan-scan rectangle information follows.

`pan_scan_cnt_minus1` specifies the number of pan-scan rectangles that are present in the SEI message. `pan_scan_cnt_minus1` shall be in the range of 0 to 2, inclusive. `pan_scan_cnt_minus1` equal to 0 indicates that a single pan-scan rectangle is present that applies to all fields of the decoded picture. `pan_scan_cnt_minus1` shall be equal to 0 when the current picture is a field. `pan_scan_cnt_minus1` equal to 1 indicates that two pan-scan rectangles are present, the first of which applies to the first field of the picture in output order and the second of which applies to the second field of the picture in output order. `pan_scan_cnt_minus1` equal to 2 indicates that three pan-scan rectangles are present, the first of which applies to the first field of the picture in output order, the second of which applies to the second field of the picture in output order, and the third of which applies to a repetition of the first field as a third field in output order.

`pan_scan_rect_left_offset[i]`, **`pan_scan_rect_right_offset[i]`**, **`pan_scan_rect_top_offset[i]`**, and **`pan_scan_rect_bottom_offset[i]`**, specify, as signed integer quantities in units of one-sixteenth sample spacing relative to the luma sampling grid, the location of the pan-scan rectangle. The values of each of these four syntax elements shall be in the range of -2^{31} to $2^{31}-1$, inclusive.

The pan-scan rectangle is specified, in units of one-sixteenth sample spacing relative to a luma frame sampling grid, as the region with frame horizontal coordinates from $16 * \text{CropUnitX} * \text{frame_crop_left_offset} + \text{pan_scan_rect_left_offset}[i]$ to $16 * (16 * \text{PicWidthInMbs} - \text{CropUnitX} * \text{frame_crop_right_offset}) + \text{pan_scan_rect_right_offset}[i] - 1$ and with vertical coordinates from $16 * \text{CropUnitY} * \text{frame_crop_top_offset} + \text{pan_scan_rect_top_offset}[i]$ to $16 * (16 * \text{PicHeightInMbs} - \text{CropUnitY} * \text{frame_crop_bottom_offset}) + \text{pan_scan_rect_bottom_offset}[i] - 1$, inclusive. The value of $16 * \text{CropUnitX} * \text{frame_crop_left_offset} + \text{pan_scan_rect_left_offset}[i]$ shall be less than or equal to $16 * (16 * \text{PicWidthInMbs} - \text{CropUnitX} * \text{frame_crop_right_offset}) + \text{pan_scan_rect_right_offset}[i] - 1$; and the value of $16 * \text{CropUnitY} * \text{frame_crop_top_offset} + \text{pan_scan_rect_top_offset}[i]$ shall be less than or equal to $16 * (16 * \text{PicHeightInMbs} - \text{CropUnitY} * \text{frame_crop_bottom_offset}) + \text{pan_scan_rect_bottom_offset}[i] - 1$.

When the pan-scan rectangular area includes samples outside of the cropping rectangle, the region outside of the cropping rectangle may be filled with synthesized content (such as black video content or neutral grey video content) for display.

`pan_scan_rect_repetition_period` specifies the persistence of the pan-scan rectangle SEI message and may specify a picture order count interval within which another pan-scan rectangle SEI message with the same value of `pan_scan_rect_id` or the end of the coded video sequence shall be present in the bitstream. The value of `pan_scan_rect_repetition_period` shall be in the range of 0 to 16 384, inclusive. When `pan_scan_cnt_minus1` is greater than 0, `pan_scan_rect_repetition_period` shall not be greater than 1.

`pan_scan_rect_repetition_period` equal to 0 specifies that the pan-scan rectangle information applies to the current decoded picture only.

`pan_scan_rect_repetition_period` equal to 1 specifies that the pan-scan rectangle information persists in output order until any of the following conditions are true.

- A new coded video sequence begins
- A picture in an access unit containing a pan-scan rectangle SEI message with the same value of `pan_scan_rect_id` is output having `PicOrderCnt()` greater than `PicOrderCnt(CurrPic)`.

`pan_scan_rect_repetition_period` equal to 0 or equal to 1 indicates that another pan-scan rectangle SEI message with the same value of `pan_scan_rect_id` may or may not be present.

`pan_scan_rect_repetition_period` greater than 1 specifies that the pan-scan rectangle information persists until any of the following conditions are true.

- A new coded video sequence begins
- A picture in an access unit containing a pan-scan rectangle SEI message with the same value of `pan_scan_rect_id` is output having `PicOrderCnt()` greater than `PicOrderCnt(CurrPic)` and less than or equal to `PicOrderCnt(CurrPic) + pan_scan_rect_repetition_period`.

`pan_scan_rect_repetition_period` greater than 1 indicates that another pan-scan rectangle SEI message with the same value of `pan_scan_rect_id` shall be present for a picture in an access unit that is output having `PicOrderCnt()` greater

than $\text{PicOrderCnt}(\text{CurrPic})$ and less than or equal to $\text{PicOrderCnt}(\text{CurrPic}) + \text{pan_scan_rect_repetition_period}$; unless the bitstream ends or a new coded video sequence begins without output of such a picture.

D.2.4 Filler payload SEI message semantics

This message contains a series of payloadSize bytes of value 0xFF, which can be discarded.

ff_byte shall be a byte having the value 0xFF.

D.2.5 User data registered by ITU-T Rec. T.35 SEI message semantics

This message contains user data registered as specified by ITU-T Rec. T.35, the contents of which are not specified by this Recommendation | International Standard.

itu_t_t35_country_code shall be a byte having a value specified as a country code by ITU-T Rec. T.35 Annex A.

itu_t_t35_country_code_extension_byte shall be a byte having a value specified as a country code by ITU-T Rec. T.35 Annex B.

itu_t_t35_payload_byte shall be a byte containing data registered as specified by ITU-T Rec. T.35.

The ITU-T T.35 terminal provider code and terminal provider oriented code shall be contained in the first one or more bytes of the **itu_t_t35_payload_byte**, in the format specified by the Administration that issued the terminal provider code. Any remaining **itu_t_t35_payload_byte** data shall be data having syntax and semantics as specified by the entity identified by the ITU-T T.35 country code and terminal provider code.

D.2.6 User data unregistered SEI message semantics

This message contains unregistered user data identified by a UUID, the contents of which are not specified by this Recommendation | International Standard.

uuid_iso_iec_11578 shall have a value specified as a UUID according to the procedures of ISO/IEC 11578:1996 Annex A.

user_data_payload_byte shall be a byte containing data having syntax and semantics as specified by the UUID generator.

D.2.7 Recovery point SEI message semantics

The recovery point SEI message assists a decoder in determining when the decoding process will produce acceptable pictures for display after the decoder initiates random access or after the encoder indicates a broken link in the sequence. When the decoding process is started with the access unit in decoding order associated with the recovery point SEI message, all decoded pictures at or subsequent to the recovery point in output order specified in this SEI message are indicated to be correct or approximately correct in content. Decoded pictures produced by random access at or before the picture associated with the recovery point SEI message need not be correct in content until the indicated recovery point, and the operation of the decoding process starting at the picture associated with the recovery point SEI message may contain references to pictures not available in the decoded picture buffer.

In addition, by use of the **broken_link_flag**, the recovery point SEI message can indicate to the decoder the location of some pictures in the bitstream that can result in serious visual artefacts when displayed, even when the decoding process was begun at the location of a previous IDR access unit in decoding order.

NOTE 1 – The **broken_link_flag** can be used by encoders to indicate the location of a point after which the decoding process for the decoding of some pictures may cause references to pictures that, though available for use in the decoding process, are not the pictures that were used for reference when the bitstream was originally encoded (e.g., due to a splicing operation performed during the generation of the bitstream).

The recovery point is specified as a count in units of **frame_num** increments subsequent to the **frame_num** of the current access unit at the position of the SEI message.

NOTE 2 – When HRD information is present in the bitstream, a buffering period SEI message should be associated with the access unit associated with the recovery point SEI message in order to establish initialisation of the HRD buffer model after a random access.

recovery_frame_cnt specifies the recovery point of output pictures in output order. All decoded pictures in output order are indicated to be correct or approximately correct in content starting at the output order position of the reference picture having the **frame_num** equal to the **frame_num** of the VCL NAL units for the current access unit incremented by **recovery_frame_cnt** in modulo **MaxFrameNum** arithmetic. **recovery_frame_cnt** shall be in the range of 0 to **MaxFrameNum** – 1, inclusive.

exact_match_flag indicates whether decoded pictures at and subsequent to the specified recovery point in output order derived by starting the decoding process at the access unit associated with the recovery point SEI message shall be an

exact match to the pictures that would be produced by starting the decoding process at the location of a previous IDR access unit in the NAL unit stream. The value 0 indicates that the match need not be exact and the value 1 indicates that the match shall be exact.

When decoding starts from the location of the recovery point SEI message, all references to not available reference pictures shall be inferred as references to pictures containing only macroblocks coded using Intra macroblock prediction modes and having sample values given by Y samples equal to 128, Cb samples equal to 128, and Cr samples equal to 128 (mid-level grey) for purposes of determining the conformance of the value of `exact_match_flag`.

NOTE 3 – When performing random access, decoders should infer all references to not available reference pictures as references to pictures containing only intra macroblocks and having sample values given by Y equal to 128, Cb equal to 128, and Cr equal to 128 (mid-level grey), regardless of the value of `exact_match_flag`.

When `exact_match_flag` is equal to 0, the quality of the approximation at the recovery point is chosen by the encoding process and is not specified by this Recommendation | International Standard.

`broken_link_flag` indicates the presence or absence of a broken link in the NAL unit stream at the location of the recovery point SEI message and is assigned further semantics as follows.

- If `broken_link_flag` is equal to 1, pictures produced by starting the decoding process at the location of a previous IDR access unit may contain undesirable visual artefacts to the extent that decoded pictures at and subsequent to the access unit associated with the recovery point SEI message in decoding order should not be displayed until the specified recovery point in output order.
- Otherwise (`broken_link_flag` is equal to 0), no indication is given regarding any potential presence of visual artefacts.

Regardless of the value of the `broken_link_flag`, pictures subsequent to the specified recovery point in output order are specified to be correct or approximately correct in content.

NOTE 4 – When a sub-sequence information SEI message is present in conjunction with a recovery point SEI message in which `broken_link_flag` is equal to 1 and when `sub_seq_layer_num` is equal to 0, `sub_seq_id` should be different from the latest `sub_seq_id` for `sub_seq_layer_num` equal to 0 that was decoded prior to the location of the recovery point SEI message. When `broken_link_flag` is equal to 0, the `sub_seq_id` in sub-sequence layer 0 should remain unchanged.

`changing_slice_group_idc` equal to 0 indicates that decoded pictures are correct or approximately correct in content at and subsequent to the recovery point in output order when all macroblocks of the primary coded pictures are decoded within the changing slice group period, i.e., the period between the access unit associated with the recovery point SEI message (inclusive) and the specified recovery point (inclusive) in decoding order. `changing_slice_group_idc` shall be equal to 0 when `num_slice_groups_minus1` is equal to 0 in any primary coded picture within the changing slice group period.

When `changing_slice_group_idc` is equal to 1 or 2, `num_slice_groups_minus1` shall be equal to 1 and the macroblock-to-slice-group map type 3, 4, or 5 shall be applied in each primary coded picture in the changing slice group period.

`changing_slice_group_idc` equal to 1 indicates that within the changing slice group period no sample values outside the decoded macroblocks covered by slice group 0 are used for inter prediction of any macroblock within slice group 0. In addition, `changing_slice_group_idc` equal to 1 indicates that when all macroblocks in slice group 0 within the changing slice group period are decoded, decoded pictures will be correct or approximately correct in content at and subsequent to the specified recovery point in output order regardless of whether any macroblock in slice group 1 within the changing slice group period is decoded.

`changing_slice_group_idc` equal to 2 indicates that within the changing slice group period no sample values outside the decoded macroblocks covered by slice group 1 are used for inter prediction of any macroblock within slice group 1. In addition, `changing_slice_group_idc` equal to 2 indicates that when all macroblocks in slice group 1 within the changing slice group period are decoded, decoded pictures will be correct or approximately correct in content at and subsequent to the specified recovery point in output order regardless of whether any macroblock in slice group 0 within the changing slice group period is decoded.

`changing_slice_group_idc` shall be in the range of 0 to 2, inclusive.

D.2.8 Decoded reference picture marking repetition SEI message semantics

The decoded reference picture marking repetition SEI message is used to repeat the decoded reference picture marking syntax structure that was located in the slice header of an earlier picture in the sequence in decoding order.

`original_idr_flag` shall be equal to 1 when the decoded reference picture marking syntax structure occurred originally in an IDR picture. `original_idr_flag` shall be equal to 0 when the repeated decoded reference picture marking syntax structure did not occur in an IDR picture originally.

original_frame_num shall be equal to the frame_num of the picture where the repeated decoded reference picture marking syntax structure originally occurred. The picture indicated by original_frame_num is the previous coded picture having the specified value of frame_num. The value of original_frame_num used to refer to a picture having a memory_management_control_operation equal to 5 shall be 0.

original_field_pic_flag shall be equal to the field_pic_flag of the picture where the repeated decoded reference picture marking syntax structure originally occurred.

original_bottom_field_flag shall be equal to the bottom_field_flag of the picture where the repeated decoded reference picture marking syntax structure originally occurred.

dec_ref_pic_marking() shall contain a copy of the decoded reference picture marking syntax structure of the picture whose frame_num was original_frame_num. The nal_unit_type used for specification of the repeated dec_ref_pic_marking() syntax structure shall be the nal_unit_type of the slice header(s) of the picture whose frame_num was original_frame_num (i.e., nal_unit_type as used in subclause 7.3.3.3 shall be considered equal to 5 when original_idr_flag is equal to 1 and shall not be considered equal to 5 when original_idr_flag is equal to 0).

D.2.9 Spare picture SEI message semantics

This SEI message indicates that certain slice group map units, called spare slice group map units, in one or more decoded reference pictures resemble the co-located slice group map units in a specified decoded picture called the target picture. A spare slice group map unit may be used to replace a co-located, incorrectly decoded slice group map unit, in the target picture. A decoded picture containing spare slice group map units is called a spare picture.

For all spare pictures identified in a spare picture SEI message, the value of frame_mbs_only_flag shall be equal to the value of frame_mbs_only_flag of the target picture in the same SEI message. The spare pictures in the SEI message are constrained as follows.

- If the target picture is a decoded field, all spare pictures identified in the same SEI message shall be decoded fields.
- Otherwise (the target picture is a decoded frame), all spare pictures identified in the same SEI message shall be decoded frames.

For all spare pictures identified in a spare picture SEI message, the values of pic_width_in_mbs_minus1 and pic_height_in_map_units_minus1 shall be equal to the values of pic_width_in_mbs_minus1 and pic_height_in_map_units_minus1, respectively, of the target picture in the same SEI message. The picture associated (as specified in subclause 7.4.1.2.3) with this message shall appear after the target picture, in decoding order.

target_frame_num indicates the frame_num of the target picture.

spare_field_flag equal to 0 indicates that the target picture and the spare pictures are decoded frames. spare_field_flag equal to 1 indicates that the target picture and the spare pictures are decoded fields.

target_bottom_field_flag equal to 0 indicates that the target picture is a top field. target_bottom_field_flag equal to 1 indicates that the target picture is a bottom field.

A target picture is a decoded reference picture whose corresponding primary coded picture precedes the current picture, in decoding order, and in which the values of frame_num, field_pic_flag (when present) and bottom_field_flag (when present) are equal to target_frame_num, spare_field_flag and target_bottom_field_flag, respectively.

num_spare_pics_minus1 indicates the number of spare pictures for the specified target picture. The number of spare pictures is equal to num_spare_pics_minus1 + 1. The value of num_spare_pics_minus1 shall be in the range of 0 to 15, inclusive.

delta_spare_frame_num[i] is used to identify the spare picture that contains the i-th set of spare slice group map units, hereafter called the i-th spare picture, as specified below. The value of delta_spare_frame_num[i] shall be in the range of 0 to MaxFrameNum - 1 - !spare_field_flag, inclusive.

The frame_num of the i-th spare picture, spareFrameNum[i], is derived as follows for all values of i from 0 to num_spare_pics_minus1, inclusive:

```

candidateSpareFrameNum = target_frame_num - !spare_field_flag
for ( i = 0; i <= num_spare_pics_minus1; i++ ) {
    if( candidateSpareFrameNum < 0 )
        candidateSpareFrameNum = MaxFrameNum - 1
    spareFrameNum[ i ] = candidateSpareFrameNum - delta_spare_frame_num[ i ]
    if( spareFrameNum[ i ] < 0 )
        spareFrameNum[ i ] = MaxFrameNum + spareFrameNum[ i ]
}

```

(D-3)

```

candidateSpareFrameNum = spareFrameNum[ i ] - !spare_field_flag
}

```

spare_bottom_field_flag[i] equal to 0 indicates that the i-th spare picture is a top field. **spare_bottom_field_flag[i]** equal to 1 indicates that the i-th spare picture is a bottom field.

The 0-th spare picture is a decoded reference picture whose corresponding primary coded picture precedes the target picture, in decoding order, and in which the values of **frame_num**, **field_pic_flag** (when present) and **bottom_field_flag** (when present) are equal to **spareFrameNum[0]**, **spare_field_flag** and **spare_bottom_field_flag[0]**, respectively. The i-th spare picture is a decoded reference picture whose corresponding primary coded picture precedes the (i - 1)-th spare picture, in decoding order, and in which the values of **frame_num**, **field_pic_flag** (when present) and **bottom_field_flag** (when present) are equal to **spareFrameNum[i]**, **spare_field_flag** and **spare_bottom_field_flag[i]**, respectively.

spare_area_idc[i] indicates the method used to identify the spare slice group map units in the i-th spare picture. **spare_area_idc[i]** shall be in the range of 0 to 2, inclusive. **spare_area_idc[i]** equal to 0 indicates that all slice group map units in the i-th spare picture are spare units. **spare_area_idc[i]** equal to 1 indicates that the value of the syntax element **spare_unit_flag[i][j]** is used to identify the spare slice group map units. **spare_area_idc[i]** equal to 2 indicates that the **zero_run_length[i][j]** syntax element is used to derive the values of **spareUnitFlagInBoxOutOrder[i][j]**, as described below.

spare_unit_flag[i][j] equal to 0 indicates that the j-th slice group map unit in raster scan order in the i-th spare picture is a spare unit. **spare_unit_flag[i][j]** equal to 1 indicates that the j-th slice group map unit in raster scan order in the i-th spare picture is not a spare unit.

zero_run_length[i][j] is used to derive the values of **spareUnitFlagInBoxOutOrder[i][j]** when **spare_area_idc[i]** is equal to 2. In this case, the spare slice group map units identified in **spareUnitFlagInBoxOutOrder[i][j]** appear in counter-clockwise box-out order, as specified in subclause 8.2.2.4, for each spare picture. **spareUnitFlagInBoxOutOrder[i][j]** equal to 0 indicates that the j-th slice group map unit in counter-clockwise box-out order in the i-th spare picture is a spare unit. **spareUnitFlagInBoxOutOrder[i][j]** equal to 1 indicates that the j-th slice group map unit in counter-clockwise box-out order in the i-th spare picture is not a spare unit.

When **spare_area_idc[0]** is equal to 2, **spareUnitFlagInBoxOutOrder[0][j]** is derived as follows:

```

for( j = 0, loop = 0; j < PicSizeInMapUnits; loop++ ) {
    for( k = 0; k < zero_run_length[ 0 ][ loop ]; k++ )
        spareUnitFlagInBoxOutOrder[ 0 ][ j++ ] = 0
        spareUnitFlagInBoxOutOrder[ 0 ][ j++ ] = 1
}

```

(D-4)

When **spare_area_idc[i]** is equal to 2 and the value of i is greater than 0, **spareUnitFlagInBoxOutOrder[i][j]** is derived as follows:

```

for( j = 0, loop = 0; j < PicSizeInMapUnits; loop++ ) {
    for( k = 0; k < zero_run_length[ i ][ loop ]; k++ )
        spareUnitFlagInBoxOutOrder[ i ][ j ] = spareUnitFlagInBoxOutOrder[ i - 1 ][ j++ ]
        spareUnitFlagInBoxOutOrder[ i ][ j ] = !spareUnitFlagInBoxOutOrder[ i - 1 ][ j++ ]
}

```

(D-5)

D.2.10 Scene information SEI message semantics

A scene and a scene transition are herein defined as a set of consecutive pictures in output order.

NOTE 1 – Decoded pictures within one scene generally have similar content. The scene information SEI message is used to label pictures with scene identifiers and to indicate scene changes. The message specifies how the source pictures for the labelled pictures were created. The decoder may use the information to select an appropriate algorithm to conceal transmission errors. For example, a specific algorithm may be used to conceal transmission errors that occurred in pictures belonging to a gradual scene transition. Furthermore, the scene information SEI message may be used in a manner determined by the application, such as for indexing the scenes of a coded sequence.

A scene information SEI message labels all pictures, in decoding order, from the primary coded picture to which the SEI message is associated (inclusive), as specified in subclause 7.4.1.2.3, to the primary coded picture to which the next scene information SEI message (if present) in decoding order is associated (exclusive) or (otherwise) to the last access unit in the bitstream (inclusive). These pictures are herein referred to as the target pictures.

scene_info_present_flag equal to 0 indicates that the scene or scene transition to which the target pictures belong is unspecified. **scene_info_present_flag** equal to 1 indicates that the target pictures belong to the same scene or scene transition.

scene_id identifies the scene to which the target pictures belong. When the value of **scene_transition_type** of the target pictures is less than 4, and the previous picture in output order is marked with a value of **scene_transition_type** less than 4, and the value of **scene_id** is the same as the value of **scene_id** of the previous picture in output order, this indicates that the source scene for the target pictures and the source scene for the previous picture (in output order) are considered by the encoder to have been the same scene. When the value of **scene_transition_type** of the target pictures is greater than 3, and the previous picture in output order is marked with a value of **scene_transition_type** less than 4, and the value of **scene_id** is the same as the value of **scene_id** of the previous picture in output order, this indicates that one of the source scenes for the target pictures and the source scene for the previous picture (in output order) are considered by the encoder to have been the same scene. When the value of **scene_id** is not equal to the value of **scene_id** of the previous picture in output order, this indicates that the target pictures and the previous picture (in output order) are considered by the encoder to have been from different source scenes.

The value of **scene_id** shall be in the range of 0 to $2^{32}-1$, inclusive. Values of **scene_id** in the range of 0 to 255, inclusive, and in the range of 512 to $2^{31}-1$, inclusive, may be used as determined by the application. Values of **scene_id** in the range of 256 to 511, inclusive, and in the range of 2^{31} to $2^{32}-1$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of **scene_id** in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32}-1$, inclusive, shall ignore (remove from the bitstream and discard) it.

scene_transition_type specifies in which type of a scene transition (if any) the target pictures are involved. The valid values of **scene_transition_type** are specified in Table D-4.

Table D-4 – scene_transition_type values

Value	Description
0	No transition
1	Fade to black
2	Fade from black
3	Unspecified transition from or to constant colour
4	Dissolve
5	Wipe
6	Unspecified mixture of two scenes

When **scene_transition_type** is greater than 3, the target pictures include contents both from the scene labelled by its **scene_id** and the next scene, in output order, which is labelled by **second_scene_id** (see below). The term “the current scene” is used to indicate the scene labelled by **scene_id**. The term “the next scene” is used to indicate the scene labelled by **second_scene_id**. It is not required for any following picture, in output order, to be labelled with **scene_id** equal to **second_scene_id** of the current SEI message.

Scene transition types are specified as follows.

“No transition” specifies that the target pictures are not involved in a gradual scene transition.

NOTE 2 – When two consecutive pictures in output order have **scene_transition_type** equal to 0 and different values of **scene_id**, a scene cut occurred between the two pictures.

“Fade to black” indicates that the target pictures are part of a sequence of pictures, in output order, involved in a fade to black scene transition, i.e., the luma samples of the scene gradually approach zero and the chroma samples of the scene gradually approach 128.

NOTE 3 – When two pictures are labelled to belong to the same scene transition and their **scene_transition_type** is “Fade to black”, the later one, in output order, is darker than the previous one.

“Fade from black” indicates that the target pictures are part of a sequence of pictures, in output order, involved in a fade from black scene transition, i.e., the luma samples of the scene gradually diverge from zero and the chroma samples of the scene may gradually diverge from 128.

NOTE 4 – When two pictures are labelled to belong to the same scene transition and their **scene_transition_type** is “Fade from black”, the later one in output order is lighter than the previous one.

“Dissolve” indicates that the sample values of each target picture (before encoding) were generated by calculating a sum of co-located weighted sample values of a picture from the current scene and a picture from the next scene. The

weight of the current scene gradually decreases from full level to zero level, whereas the weight of the next scene gradually increases from zero level to full level. When two pictures are labelled to belong to the same scene transition and their `scene_transition_type` is "Dissolve", the weight of the current scene for the later one, in output order, is less than the weight of the current scene for the previous one, and the weight of the next scene for the later one, in output order, is greater than the weight of the next scene for the previous one.

"Wipe" indicates that some of the sample values of each target picture (before encoding) were generated by copying co-located sample values of a picture in the current scene and the remaining sample values of each target picture (before encoding) were generated by copying co-located sample values of a picture in the next scene. When two pictures are labelled to belong to the same scene transition and their `scene_transition_type` is "Wipe", the number of samples copied from the next scene to the later picture in output order is greater than the number of samples copied from the next scene to the previous picture.

second_scene_id identifies the next scene in the gradual scene transition in which the target pictures are involved. The value of `second_scene_id` shall not be equal to the value of `scene_id`. The value of `second_scene_id` shall not be equal to the value of `scene_id` in the previous picture in output order. When the next picture in output order is marked with a value of `scene_transition_type` less than 4, and the value of `second_scene_id` is the same as the value of `scene_id` of the next picture in output order, this indicates that the encoder considers one of the source scenes for the target pictures and the source scene for the next picture (in output order) to have been the same scene. When the value of `second_scene_id` is not equal to the value of `scene_id` or `second_scene_id` (if present) of the next picture in output order, this indicates that the encoder considers the target pictures and the next picture (in output order) to have been from different source scenes.

When the value of `scene_id` of a picture is equal to the value of `scene_id` of the following picture in output order and the value of `scene_transition_type` in both of these pictures is less than 4, this indicates that the encoder considers the two pictures to have been from the same source scene. When the values of `scene_id`, `scene_transition_type` and `second_scene_id` (if present) of a picture are equal to the values of `scene_id`, `scene_transition_type` and `second_scene_id` (respectively) of the following picture in output order and the value of `scene_transition_type` is greater than 0, this indicates that the encoder considers the two pictures to have been from the same source gradual scene transition.

The value of `second_scene_id` shall be in the range of 0 to $2^{32}-1$, inclusive. Values of `second_scene_id` in the range of 0 to 255, inclusive, and in the range of 512 to $2^{31}-1$, inclusive, may be used as determined by the application. Values of `second_scene_id` in the range of 256 to 511, inclusive, and in the range of 2^{31} to $2^{32}-1$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of `second_scene_id` in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32}-1$, inclusive, shall ignore (remove from the bitstream and discard) it.

D.2.11 Sub-sequence information SEI message semantics

The sub-sequence information SEI message is used to indicate the position of a picture in data dependency hierarchy that consists of sub-sequence layers and sub-sequences.

A sub-sequence layer contains a subset of the coded pictures in a sequence. Sub-sequence layers are numbered with non-negative integers. A layer having a larger layer number is a higher layer than a layer having a smaller layer number. The layers are ordered hierarchically based on their dependency on each other so that any picture in a layer shall not be predicted from any picture on any higher layer.

NOTE 1 – In other words, any picture in layer 0 must not be predicted from any picture in layer 1 or above, pictures in layer 1 may be predicted from layer 0, pictures in layer 2 may be predicted from layers 0 and 1, etc.

NOTE 2 – The subjective quality is expected to increase along with the number of decoded layers.

A sub-sequence is a set of coded pictures within a sub-sequence layer. A picture shall reside in one sub-sequence layer and in one sub-sequence only. Any picture in a sub-sequence shall not be predicted from any picture in another sub-sequence in the same or in a higher sub-sequence layer. A sub-sequence in layer 0 can be decoded independently of any picture that does not belong to the sub-sequence.

The sub-sequence information SEI message concerns the current access unit. The primary coded picture in the access unit is herein referred to as the current picture.

The sub-sequence information SEI message shall not be present unless `gaps_in_frame_num_value_allowed_flag` in the sequence parameter set referenced by the picture associated with the sub-sequence SEI message is equal to 1.

sub_seq_layer_num specifies the sub-sequence layer number of the current picture. When `sub_seq_layer_num` is greater than 0, memory management control operations shall not be used in any slice header of the current picture. When the current picture resides in a sub-sequence whose first picture in decoding order is an IDR picture, the value of `sub_seq_layer_num` shall be equal to 0. For a non-paired reference field, the value of `sub_seq_layer_num` shall be equal to 0. `sub_seq_layer_num` shall be in the range of 0 to 255, inclusive.

sub_seq_id identifies the sub-sequence within a layer. When the current picture resides in a sub-sequence whose first picture in decoding order is an IDR picture, the value of **sub_seq_id** shall be the same as the value of **idr_pic_id** of the IDR picture. **sub_seq_id** shall be in the range of 0 to 65535, inclusive.

first_ref_pic_flag equal to 1 specifies that the current picture is the first reference picture of the sub-sequence in decoding order. When the current picture is not the first picture of the sub-sequence in decoding order, the **first_ref_pic_flag** shall be equal to 0.

leading_non_ref_pic_flag equal to 1 specifies that the current picture is a non-reference picture preceding any reference picture in decoding order within the sub-sequence or that the sub-sequence contains no reference pictures. When the current picture is a reference picture or the current picture is a non-reference picture succeeding at least one reference picture in decoding order within the sub-sequence, the **leading_non_ref_pic_flag** shall be equal to 0.

last_pic_flag equal to 1 indicates that the current picture is the last picture of the sub-sequence (in decoding order), including all reference and non-reference pictures of the sub-sequence. When the current picture is not the last picture of the sub-sequence (in decoding order), **last_pic_flag** shall be equal to 0.

The current picture is assigned to a sub-sequence as follows.

- If one or more of the following conditions is true, the current picture is the first picture of a sub-sequence in decoding order.
 - no earlier picture in decoding order is labelled with the same values of **sub_seq_id** and **sub_seq_layer_num** as the current picture
 - the value of **leading_non_ref_pic_flag** is equal to 1 and the value of **leading_non_ref_pic_flag** is equal to 0 in the previous picture in decoding order having the same values of **sub_seq_id** and **sub_seq_layer_num** as the current picture
 - the value of **first_ref_pic_flag** is equal to 1 and the value of **leading_non_ref_pic_flag** is equal to 0 in the previous picture in decoding order having the same values of **sub_seq_id** and **sub_seq_layer_num** as the current picture
 - the value of **last_pic_flag** is equal to 1 in the previous picture in decoding order having the same values of **sub_seq_id** and **sub_seq_layer_num** as the current picture
- Otherwise, the current picture belongs to the same sub-sequence as the previous picture in decoding order having the same values of **sub_seq_id** and **sub_seq_layer_num** as the current picture.

sub_seq_frame_num_flag equal to 0 specifies that **sub_seq_frame_num** is not present. **sub_seq_frame_num_flag** equal to 1 specifies that **sub_seq_frame_num** is present.

sub_seq_frame_num shall be equal to 0 for the first reference picture of the sub-sequence and for any non-reference picture preceding the first reference picture of the sub-sequence in decoding order. **sub_seq_frame_num** is further constrained as follows.

- If the current picture is not the second field of a complementary field pair, **sub_seq_frame_num** shall be incremented by 1, in modulo **MaxFrameNum** operation, relative to the previous reference picture, in decoding order, that belongs to the sub-sequence.
- Otherwise (the current picture is the second field of a complementary field pair), the value of **sub_seq_frame_num** shall be the same as the value of **sub_seq_frame_num** for the first field of the complementary field pair.

sub_seq_frame_num shall be in the range of 0 to **MaxFrameNum** – 1, inclusive.

When the current picture is an IDR picture, it shall start a new sub-sequence in sub-sequence layer 0. Thus, the **sub_seq_layer_num** shall be 0, the **sub_seq_id** shall be different from the previous sub-sequence in sub-sequence layer 0, **first_ref_pic_flag** shall be 1, and **leading_non_ref_pic_flag** shall be equal to 0.

When the sub-sequence information SEI message is present for both coded fields of a complementary field pair, the values of **sub_seq_layer_num**, **sub_seq_id**, **leading_non_ref_pic_flag** and **sub_seq_frame_num**, when present, shall be identical for both of these pictures.

When the sub-sequence information SEI message is present only for one coded field of a complementary field pair, the values of **sub_seq_layer_num**, **sub_seq_id**, **leading_non_ref_pic_flag** and **sub_seq_frame_num**, when present, are also applicable to the other coded field of the complementary field pair.

D.2.12 Sub-sequence layer characteristics SEI message semantics

The sub-sequence layer characteristics SEI message specifies the characteristics of sub-sequence layers.

num_sub_seq_layers_minus1 plus 1 specifies the number of sub-sequence layers in the sequence. **num_sub_seq_layers_minus1** shall be in the range of 0 to 255, inclusive.

A pair of **average_bit_rate** and **average_frame_rate** characterizes each sub-sequence layer. The first pair of **average_bit_rate** and **average_frame_rate** specifies the characteristics of sub-sequence layer 0. When present, the second pair specifies the characteristics of sub-sequence layers 0 and 1 jointly. Each pair in decoding order specifies the characteristics for a range of sub-sequence layers from layer number 0 to the layer number specified by the layer loop counter. The values are in effect from the point they are decoded until an update of the values is decoded.

accurate_statistics_flag equal to 1 indicates that the values of **average_bit_rate** and **average_frame_rate** are rounded from statistically correct values. **accurate_statistics_flag** equal to 0 indicates that the **average_bit_rate** and the **average_frame_rate** are estimates and may deviate somewhat from the correct values.

When **accurate_statistics_flag** is equal to 0, the quality of the approximation used in the computation of the values of **average_bit_rate** and the **average_frame_rate** is chosen by the encoding process and is not specified by this Recommendation | International Standard.

average_bit_rate indicates the average bit rate in units of 1000 bits per second. All NAL units in the range of sub-sequence layers specified above are taken into account in the calculation. The average bit rate is derived according to the access unit removal time specified in Annex C of the Recommendation | International Standard. In the following, **bTotal** is the number of bits in all NAL units succeeding a sub-sequence layer characteristics SEI message (including the bits of the NAL units of the current access unit) and preceding the next access unit (in decoding order) including a sub-sequence layer characteristics SEI message (if present) or the end of the stream (otherwise). t_1 is the removal time (in seconds) of the current access unit, and t_2 is the removal time (in seconds) of the latest access unit (in decoding order) before the next sub-sequence layer characteristics SEI message (if present) or the end of the stream (otherwise).

When **accurate_statistics_flag** is equal to 1, the following conditions shall be fulfilled as follows.

- If t_1 is not equal to t_2 , the following condition shall be true

$$\text{average_bit_rate} == \text{Round}(\text{bTotal} \div ((t_2 - t_1) * 1000)) \quad (\text{D-6})$$

- Otherwise (t_1 is equal to t_2), the following condition shall be true

$$\text{average_bit_rate} == 0 \quad (\text{D-7})$$

average_frame_rate indicates the average frame rate in units of frames/(256 seconds). All NAL units in the range of sub-sequence layers specified above are taken into account in the calculation. In the following, **fTotal** is the number of frames, complementary field pairs and non-paired fields between the current picture (inclusive) and the next sub-sequence layer characteristics SEI message (if present) or the end of the stream (otherwise). t_1 is the removal time (in seconds) of the current access unit, and t_2 is the removal time (in seconds) of the latest access unit (in decoding order) before the next sub-sequence layer characteristics SEI message (if present) or the end of the stream (otherwise).

When **accurate_statistics_flag** is equal to 1, the following conditions shall be fulfilled as follows.

- If t_1 is not equal to t_2 , the following condition shall be true

$$\text{average_frame_rate} == \text{Round}(\text{fTotal} * 256 \div (t_2 - t_1)) \quad (\text{D-8})$$

- Otherwise (t_1 is equal to t_2), the following condition shall be true

$$\text{average_frame_rate} == 0 \quad (\text{D-9})$$

D.2.13 Sub-sequence characteristics SEI message semantics

The sub-sequence characteristics SEI message indicates the characteristics of a sub-sequence. It also indicates inter prediction dependencies between sub-sequences. This message shall be contained in the first access unit in decoding order of the sub-sequence to which the sub-sequence characteristics SEI message applies. This sub-sequence is herein called the target sub-sequence.

sub_seq_layer_num identifies the sub-sequence layer number of the target sub-sequence. **sub_seq_layer_num** shall be in the range of 0 to 255, inclusive.

sub_seq_id identifies the target sub-sequence. **sub_seq_id** shall be in the range of 0 to 65535, inclusive.

duration_flag equal to 0 indicates that the duration of the target sub-sequence is not specified.

sub_seq_duration specifies the duration of the target sub-sequence in clock ticks of a 90-kHz clock.

average_rate_flag equal to 0 indicates that the average bit rate and the average frame rate of the target sub-sequence are unspecified.

accurate_statistics_flag indicates how reliable the values of **average_bit_rate** and **average_frame_rate** are. **accurate_statistics_flag** equal to 1, indicates that the **average_bit_rate** and the **average_frame_rate** are rounded from statistically correct values. **accurate_statistics_flag** equal to 0 indicates that the **average_bit_rate** and the **average_frame_rate** are estimates and may deviate from the statistically correct values.

average_bit_rate indicates the average bit rate in (1000 bits)/second of the target sub-sequence. All NAL units of the target sub-sequence are taken into account in the calculation. The average bit rate is derived according to the access unit removal time specified in subclause C.1.2. In the following, nB is the number of bits in all NAL units in the sub-sequence. t_1 is the removal time (in seconds) of the first access unit of the sub-sequence (in decoding order), and t_2 is the removal time (in seconds) of the last access unit of the sub-sequence (in decoding order).

When **accurate_statistics_flag** is equal to 1, the following conditions shall be fulfilled as follows.

- If t_1 is not equal to t_2 , the following condition shall be true

$$\text{average_bit_rate} == \text{Round}(nB \div ((t_2 - t_1) * 1000)) \quad (D-10)$$

- Otherwise (t_1 is equal to t_2), the following condition shall be true

$$\text{average_bit_rate} == 0 \quad (D-11)$$

average_frame_rate indicates the average frame rate in units of frames/(256 seconds) of the target sub-sequence. All NAL units of the target sub-sequence are taken into account in the calculation. The average frame rate is derived according to the access unit removal time specified in subclause C.1.2. In the following, fC is the number of frames, complementary field pairs and non-paired fields in the sub-sequence. t_1 is the removal time (in seconds) of the first access unit of the sub-sequence (in decoding order), and t_2 is the removal time (in seconds) of the last access unit of the sub-sequence (in decoding order).

When **accurate_statistics_flag** is equal to 1, the following conditions shall be fulfilled as follows.

- If t_1 is not equal to t_2 , the following condition shall be true

$$\text{average_frame_rate} == \text{Round}(fC * 256 \div (t_2 - t_1)) \quad (D-12)$$

- Otherwise (t_1 is equal to t_2), the following condition shall be true

$$\text{average_frame_rate} == 0 \quad (D-13)$$

num_referenced_subseqs specifies the number of sub-sequences that contain pictures that are used as reference pictures for inter prediction in the pictures of the target sub-sequence. **num_referenced_subseqs** shall be in the range of 0 to 255, inclusive.

ref_sub_seq_layer_num, **ref_sub_seq_id**, and **ref_sub_seq_direction** identify the sub-sequence that contains pictures that are used as reference pictures for inter prediction in the pictures of the target sub-sequence. Depending on **ref_sub_seq_direction**, the following applies.

- If **ref_sub_seq_direction** is equal to 0, a set of candidate sub-sequences consists of the sub-sequences whose **sub_seq_id** is equal to **ref_sub_seq_id**, which reside in the sub-sequence layer having **sub_seq_layer_num** equal to **ref_sub_seq_layer_num**, and whose first picture in decoding order precedes the first picture of the target sub-sequence in decoding order.
- Otherwise (**ref_sub_seq_direction** is equal to 1), a set of candidate sub-sequences consists of the sub-sequences whose **sub_seq_id** is equal to **ref_sub_seq_id**, which reside in the sub-sequence layer having **sub_seq_layer_num** equal to **ref_sub_seq_layer_num**, and whose first picture in decoding order succeeds the first picture of the target sub-sequence in decoding order.

The sub-sequence used as a reference for the target sub-sequence is the sub-sequence among the set of candidate sub-sequences whose first picture is the closest to the first picture of the target sub-sequence in decoding order.

D.2.14 Full-frame freeze SEI message semantics

The full-frame freeze SEI message indicates that the current picture and any subsequent pictures in output order that meet specified conditions should not affect the content of the display. No more than one full-frame freeze SEI message shall be present in any access unit.

full_frame_freeze_repetition_period specifies the persistence of the full-frame freeze SEI message and may specify a picture order count interval within which another full-frame freeze SEI message or a full-frame freeze release SEI or the end of the coded video sequence shall be present in the bitstream. The value of **full_frame_freeze_repetition_period** shall be in the range of 0 to 16 384, inclusive.

full_frame_freeze_repetition_period equal to 0 specifies that the full-frame freeze SEI message applies to the current decoded picture only.

full_frame_freeze_repetition_period equal to 1 specifies that the full-frame freeze SEI message persists in output order until any of the following conditions are true.

- A new coded video sequence begins
- A picture in an access unit containing a full-frame freeze SEI message or a full-frame freeze release SEI message is output having **PicOrderCnt()** greater than **PicOrderCnt(CurrPic)**.

full_frame_freeze_repetition_period greater than 1 specifies that the full-frame freeze SEI message persists until any of the following conditions are true.

- A new coded video sequence begins
- A picture in an access unit containing a full-frame freeze SEI message or a full-frame freeze release SEI message is output having **PicOrderCnt()** greater than **PicOrderCnt(CurrPic)** and less than or equal to **PicOrderCnt(CurrPic) + full_frame_freeze_repetition_period**.

full_frame_freeze_repetition_period greater than 1 indicates that another full-frame freeze SEI message or a full-frame freeze release SEI message shall be present for a picture in an access unit that is output having **PicOrderCnt()** greater than **PicOrderCnt(CurrPic)** and less than or equal to **PicOrderCnt(CurrPic) + full_frame_freeze_repetition_period**; unless the bitstream ends or a new coded video sequence begins without output of such a picture.

D.2.15 Full-frame freeze release SEI message semantics

The full-frame freeze release SEI message cancels the effect of any full-frame freeze SEI message sent with pictures that precede the current picture in output order. The full-frame freeze release SEI message indicates that the current picture and subsequent pictures in output order should affect the contents of the display.

No more than one full-frame freeze release SEI message shall be present in any access unit. A full-frame freeze release SEI message shall not be present in an access unit containing a full-frame freeze SEI message. When a full-frame freeze SEI message is present in an access unit containing a field of a complementary field pair in which the values of **PicOrderCnt(CurrPic)** for the two fields of the complementary field pair are equal to each other, a full-frame freeze release SEI message shall not be present in either of the two access units.

D.2.16 Full-frame snapshot SEI message semantics

The full-frame snapshot SEI message indicates that the current frame is labelled for use as determined by the application as a still-image snapshot of the video content.

snapshot_id specifies a snapshot identification number. **snapshot_id** shall be in the range of 0 to $2^{32} - 1$, inclusive.

Values of **snapshot_id** in the range of 0 to 255, inclusive, and in the range of 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of **snapshot_id** in the range of 256 to 511, inclusive, and in the range of 2^{31} to $2^{32} - 1$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of **snapshot_id** in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 1$, inclusive, shall ignore (remove from the bitstream and discard) it.

D.2.17 Progressive refinement segment start SEI message semantics

The progressive refinement segment start SEI message specifies the beginning of a set of consecutive coded pictures that is labelled as the current picture followed by a sequence of one or more pictures of refinement of the quality of the current picture, rather than as a representation of a continually moving scene.

The tagged set of consecutive coded pictures shall continue until one of the following conditions is true. When a condition below becomes true, the next slice to be decoded does not belong to the tagged set of consecutive coded pictures.

1. The next slice to be decoded belongs to an IDR picture.

2. `num_refinement_steps_minus1` is greater than 0 and the `frame_num` of the next slice to be decoded is $(\text{currFrameNum} + \text{num_refinement_steps_minus1} + 1) \% \text{MaxFrameNum}$, where `currFrameNum` is the value of `frame_num` of the picture in the access unit containing the SEI message.
3. `num_refinement_steps_minus1` is 0 and a progressive refinement segment end SEI message with the same `progressive_refinement_id` as the one in this SEI message is decoded.

The decoding order of picture within the tagged set of consecutive pictures should be the same as their output order. **progressive_refinement_id** specifies an identification number for the progressive refinement operation. `progressive_refinement_id` shall be in the range of 0 to $2^{32} - 1$, inclusive.

Values of `progressive_refinement_id` in the range of 0 to 255, inclusive, and in the range of 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of `progressive_refinement_id` in the range of 256 to 511, inclusive, and in the range of 2^{31} to $2^{32} - 1$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of `progressive_refinement_id` in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 1$, inclusive, shall ignore (remove from the bitstream and discard) it.

num_refinement_steps_minus1 specifies the number of reference frames in the tagged set of consecutive coded pictures as follows.

- If `num_refinement_steps_minus1` is equal to 0, the number of reference frames in the tagged set of consecutive coded pictures is unknown.
- Otherwise, the number of reference frames in the tagged set of consecutive coded pictures is equal to `num_refinement_steps_minus1 + 1`.

`num_refinement_steps_minus1` shall be in the range of 0 to `MaxFrameNum - 1`, inclusive.

D.2.18 Progressive refinement segment end SEI message semantics

The progressive refinement segment end SEI message specifies the end of a set of consecutive coded pictures that has been labelled by use of a progressive refinement segment start SEI message as an initial picture followed by a sequence of one or more pictures of the refinement of the quality of the initial picture, and ending with the current picture.

progressive_refinement_id specifies an identification number for the progressive refinement operation. `progressive_refinement_id` shall be in the range of 0 to $2^{32} - 1$, inclusive.

The progressive refinement segment end SEI message specifies the end of any progressive refinement segment previously started using a progressive refinement segment start SEI message with the same value of `progressive_refinement_id`.

Values of `progressive_refinement_id` in the range of 0 to 255, inclusive, and in the range of 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of `progressive_refinement_id` in the range of 256 to 511, inclusive, and in the range of 2^{31} to $2^{32} - 1$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of `progressive_refinement_id` in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 1$, inclusive, shall ignore (remove from the bitstream and discard) it.

D.2.19 Motion-constrained slice group set SEI message semantics

This SEI message indicates that inter prediction over slice group boundaries is constrained as specified below. When present, the message shall only appear where it is associated, as specified in subclause 7.4.1.2.3, with an IDR access unit.

The target picture set for this SEI message contains all consecutive primary coded pictures in decoding order starting with the associated primary coded IDR picture (inclusive) and ending with the following primary coded IDR picture (exclusive) or with the very last primary coded picture in the bitstream (inclusive) in decoding order when there is no following primary coded IDR picture. The slice group set is a collection of one or more slice groups, identified by the `slice_group_id[i]` syntax element.

This SEI message indicates that, for each picture in the target picture set, the inter prediction process is constrained as follows: No sample value outside the slice group set, and no sample value at a fractional sample position that is derived using one or more sample values outside the slice group set is used to inter predict any sample within the slice group set.

num_slice_groups_in_set_minus1 + 1 specifies the number of slice groups in the slice group set. The allowed range of `num_slice_groups_in_set_minus1` is 0 to `num_slice_groups_minus1`, inclusive. The allowed range of `num_slice_groups_minus1` is specified in Annex A.

slice_group_id[i] identifies the slice group(s) contained within the slice group set. The allowed range is from 0 to num_slice_groups_in_set_minus1, inclusive. The size of the slice_group_id[i] syntax element is $\text{Ceil}(\text{Log2}(\text{num_slice_groups_minus1} + 1))$ bits.

exact_sample_value_match_flag equal to 0 indicates that, within the target picture set, when the macroblocks that do not belong to the slice group set are not decoded, the value of each sample in the slice group set need not be exactly the same as the value of the same sample when all the macroblocks are decoded. **exact_sample_value_match_flag** equal to 1 indicates that, within the target picture set, when the macroblocks that do not belong to the slice group set are not decoded, the value of each sample in the slice group set shall be exactly the same as the value of the same sample when all the macroblocks in the target picture set are decoded.

NOTE 1 – When **disable_deblocking_filter_idc** is equal to 2 in all slices in the target picture set, **exact_sample_value_match_flag** should be 1.

pan_scan_rect_flag equal to 0 specifies that **pan_scan_rect_id** is not present. **pan_scan_rect_flag** equal to 1 specifies that **pan_scan_rect_id** is present.

pan_scan_rect_id indicates that the specified slice group set covers at least the pan-scan rectangle identified by **pan_scan_rect_id** within the target picture set.

NOTE 2 – Multiple **motion_constrained_slice_group_set** SEI messages may be associated with the same IDR picture. Consequently, more than one slice group set may be active within a target picture set.

NOTE 3 – The size, shape, and location of the slice groups in the slice group set may change within the target picture set.

D.2.20 Film grain characteristics SEI message semantics

This SEI message provides the decoder with a parameterised model for film grain synthesis. For example, an encoder may use the film grain characteristics SEI message to characterise film grain that was present in the original source video material and was removed by pre-processing filtering techniques. Synthesis of simulated film grain on the decoded images for the display process is optional and does not affect the decoding process specified in this Recommendation | International Standard. If synthesis of simulated film grain on the decoded images for the display process is performed, there is no requirement that the method by which the synthesis is performed be the same as the parameterised model for the film grain as provided in the film grain characteristics SEI message.

NOTE 1 – The display process is not specified in this Recommendation | International Standard.

film_grain_characteristics_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous film grain characteristics SEI message in output order. **film_grain_characteristics_cancel_flag** equal to 0 indicates that film grain modelling information follows.

model_id identifies the film grain simulation model as specified in Table D-5. The value of **model_id** shall be in the range of 0 to 1, inclusive.

Table D-5 – model_id values

Value	Description
0	frequency filtering
1	auto-regression
2	reserved
3	reserved

separate_colour_description_present_flag equal to 1 indicates that a distinct colour space description for the film grain characteristics specified in the SEI message is present in the film grain characteristics SEI message syntax. **separate_colour_description_present_flag** equal to 0 indicates that the colour description for the film grain characteristics specified in the SEI message is the same as for the coded video sequence as specified in subclause E.2.1.

NOTE 2 – When **separate_colour_description_present_flag** is equal to 1, the colour space specified for the film grain characteristics specified in the SEI message may differ from the colour space specified for the coded video as specified in subclause E.2.1.

film_grain_bit_depth_luma_minus8 plus 8 specifies the bit depth used for the luma component of the film grain characteristics specified in the SEI message. When **film_grain_bit_depth_luma_minus8** is not present in the film grain characteristics SEI message, the value of **film_grain_bit_depth_luma_minus8** shall be inferred to be equal to **bit_depth_luma_minus8**.

The value of **filmGrainBitDepth[0]** is derived as

$$\text{filmGrainBitDepth}[0] = \text{film_grain_bit_depth_luma_minus8} + 8 \quad (\text{D-14})$$

film_grain_bit_depth_chroma_minus8 plus 8 specifies the bit depth used for the Cb and Cr components of the film grain characteristics specified in the SEI message. When **film_grain_bit_depth_chroma_minus8** is not present in the film grain characteristics SEI message, the value of **film_grain_bit_depth_chroma_minus8** shall be inferred to be equal to **bit_depth_chroma_minus8**.

The value of **filmGrainBitDepth[c]** for **c = 1** and **2** is derived as

$$\text{filmGrainBitDepth}[c] = \text{film_grain_bit_depth_chroma_minus8} + 8 \quad \text{with } c = 1, 2 \quad (\text{D-15})$$

film_grain_full_range_flag has the same semantics as specified in subclause E.2.1 for the **video_full_range_flag** syntax element, except as follows.

- **film_grain_full_range_flag** specifies the colour space of the film grain characteristics specified in the SEI message, rather than the colour space used for the coded video sequence.
- When **film_grain_full_range_flag** is not present in the film grain characteristics SEI message, the value of **film_grain_full_range_flag** shall be inferred to be equal to **video_full_range_flag**.

film_grain_colour_primaries has the same semantics as specified in subclause E.2.1 for the **colour_primaries** syntax element, except as follows.

- **film_grain_colour_primaries** specifies the colour space of the film grain characteristics specified in the SEI message, rather than the colour space used for the coded video sequence.
- When **film_grain_colour_primaries** is not present in the film grain characteristics SEI message, the value of **film_grain_colour_primaries** shall be inferred to be equal to **colour_primaries**.

film_grain_transfer_characteristics has the same semantics as specified in subclause E.2.1 for the **transfer_characteristics** syntax element, except as follows.

- **film_grain_transfer_characteristics** specifies the colour space of the film grain characteristics specified in the SEI message, rather than the colour space used for the coded video sequence.
- When **film_grain_transfer_characteristics** is not present in the film grain characteristics SEI message, the value of **film_grain_transfer_characteristics** shall be inferred to be equal to **transfer_characteristics**.

film_grain_matrix_coefficients has the same semantics as specified in subclause E.2.1 for the **matrix_coefficients** syntax element, except as follows.

- **film_grain_matrix_coefficients** specifies the colour space of the film grain characteristics specified in the SEI message, rather than the colour space used for the coded video sequence.
- When **film_grain_matrix_coefficients** is not present in the film grain characteristics SEI message, the value of **film_grain_matrix_coefficients** shall be inferred to be equal to **matrix_coefficients**.
- The values allowed for **film_grain_matrix_coefficients** are not constrained by the value of **chroma_format_idc**.

The **chroma_format_idc** of the film grain characteristics specified in the film grain characteristics SEI message shall be inferred to be equal to 3 (4:4:4).

NOTE 3 – Because the use of a specific method is not required for performing film grain generation function used by the display process, a decoder may, if desired, down-convert the model information for chroma in order to simulate film grain for other chroma formats (4:2:0 or 4:2:2) rather than up-converting the decoded video (using a method not specified by this Recommendation | International Standard) before performing film grain generation.

blending_mode_id identifies the blending mode used to blend the simulated film grain with the decoded images as specified in Table D-6. **blending_mode_id** shall be in the range of 0 to 1, inclusive.

Table D-6 – blending_mode_id values

Value	Description
0	additive
1	multiplicative
2	reserved
3	reserved

Depending on `blending_mode_id`, the blending mode is specified as follows

- If `blending_mode_id` is equal to 0 the blending mode is additive as specified by

$$I_{\text{grain}}[x, y, c] = \text{Clip3}(0, (1 \ll \text{filmGrainBitDepth}[c]) - 1, I_{\text{decoded}}[x, y, c] + G[x, y, c]) \quad (\text{D-16})$$

- Otherwise (`blending_mode_id` is equal to 1), the blending mode is multiplicative as specified by

$$I_{\text{grain}}[x, y, c] = \text{Clip3}(0, (1 \ll \text{filmGrainBitDepth}[c]) - 1, I_{\text{decoded}}[x, y, c] * (1 + G[x, y, c])) \quad (\text{D-17})$$

where $I_{\text{decoded}}[x, y, c]$ represents the sample value at coordinates x, y of the colour component c of the decoded image I_{decoded} , $G[x, y, c]$ is the simulated film grain value at the same position and colour component, and $\text{filmGrainBitDepth}[c]$ is the number of bits used for each sample in a fixed-length unsigned binary representation of the array $I_{\text{grain}}[x, y, c]$.

log2_scale_factor specifies a scale factor used in the film grain characterization equations.

comp_model_present_flag[c] equal to 0 indicates that film grain is not modelled on the c -th colour component, where c equal to 0 refers to the luma component, c equal to 1 refers to the Cb component, and c equal to 2 refers to the Cr component. **comp_model_present_flag**[c] equal to 1 indicates that syntax elements specifying modelling of film grain on colour component c are present in the SEI message.

num_intensity_intervals_minus1[c] plus 1 specifies the number of intensity intervals for which a specific set of model values has been estimated.

NOTE 4 – The intensity intervals may overlap in order to simulate multi-generational film grain.

num_model_values_minus1[c] plus 1 specifies the number of model values present for each intensity interval in which the film grain has been modelled. The value of **num_model_values_minus1**[c] shall be in the range of 0 to 5, inclusive.

intensity_interval_lower_bound[c][i] specifies the lower bound of the interval i of intensity levels for which the set of model values applies.

intensity_interval_upper_bound[c][i] specifies the upper bound of the interval i of intensity levels for which the set of model values applies.

Depending on `model_id`, the selection of the sets of model values is specified as follows.

- If `model_id` is equal to 0, the average value of each block b of 16x16 samples in I_{decoded} , referred as b_{avg} , is used to select the sets of model values with index $s[j]$ that apply to all the samples in the block:

```
for( i = 0, j = 0; i <= num_intensity_intervals_minus1; i++ )
    if( b_avg >= intensity_interval_lower_bound[ c ][ i ] && b_avg <= intensity_interval_upper_bound[ c ][ i ] ) {
        s[ j ] = i
        j++
    }
```

(D-18)

- Otherwise (`model_id` is equal to 1), the sets of model values used to generate the film grain are selected for each sample value in I_{decoded} as follows:

```
for( i = 0, j = 0; i <= num_intensity_intervals_minus1; i++ )
    if( I_decoded[ x, y, c ] >= intensity_interval_lower_bound[ c ][ i ] &&
        I_decoded[ x, y, c ] <= intensity_interval_upper_bound[ c ][ i ] ) {
        s[ j ] = i
        j++
    }
```

(D-19)

Samples that do not fall into any of the defined intervals are not modified by the grain generation function. Samples that fall into more than one interval will originate multi-generation grain. Multi-generation grain results from adding the grain computed independently for each intensity interval.

NOTE 6 – Coded model values are based on blocks of 16x16, but a decoder implementation may use other block sizes. For example, decoders implementing the IDCT on 8x8 blocks, should down-convert by a factor of two the set of coded model values comp_model_value[c][s][i] for i equal to 1..4.

NOTE 7 – To reduce the degree of visible blocks that can result from mosaicing the frequency-filtered blocks filteredRV, decoders may apply a low-pass filter to the boundaries between frequency-filtered blocks.

Q[c] is formed by the frequency-filtered blocks filteredRV.

where Cos(x) is the trigonometric cosine function operating on an argument x in units of radians and π is Archimedes' constant 3.141 592 653.

$$r_{ij} = \frac{4}{((i == 0) ? 1 : \sqrt{2})} \cos\left(\frac{i * (2 * j + 1) * \pi}{32}\right) \quad (\text{D-24})$$

where the superscript T indicates a matrix transposition and r is the 16x16 matrix with elements r_{ij} specified by

$$\text{IDCT16x16}(z) = r * z * r^T \quad (\text{D-23})$$

where IDCT16x16(z) refers to a unitary inverse discrete cosine transformation (IDCT) operating on a 16x16 matrix argument z as specified by

$$\begin{aligned} \text{filteredRV} &= \text{IDCT16x16}(\text{gaussRV}) \\ \text{gaussRV}[x, y] &= 0 \\ \text{if}((x < \text{comp_model_value}[c][s][3] \ \&\& \ y < \text{comp_model_value}[c][s][4]) \ || \\ \text{for}(x = 0; x < 16; x++) \\ \text{for}(y = 0; y < 16; y++) \end{aligned} \quad (\text{D-22})$$

follows:

The band-pass filtering of blocks gaussRV may be performed in the discrete cosine transform (DCT) domain as follows:

where Ln(x) is the natural logarithm of x (the base-e logarithm, where e is natural logarithm base constant 2.718 281 828...), Cos(x) is the trigonometric cosine function operating on an argument x in units of radians, and π is Archimedes' constant 3.141 592 653....

$$\text{gaussRV}_{ij} = \sqrt{-2 * \text{Ln}(\text{uRV}_0) * \text{Cos}(2 * \pi * \text{uRV}_i)} \quad (\text{D-21})$$

NOTE 5 – A normalized Gaussian random value can be generated from two independent, uniformly distributed random values over the interval from 0 to 1 (and not equal to 0), denoted as uRV₀ and uRV₁, using the Box-Muller transformation specified by

where Q[c] is a two-dimensional random process generated by filtering 16x16 blocks gaussRV with random-value elements gaussRV_{ij} generated with a normalized Gaussian distribution (independent and identically distributed Gaussian random variable samples with zero mean and unity variance) and where the value of an element G[x, y, c-1] used in the right-hand side of the equation is inferred to be equal to 0 when c-1 is less than 0.

$$G[x, y, c] = (\text{comp_model_value}[c][s][0] * Q[x, y, c] + \text{comp_model_value}[c][s][5]) * \log_2\text{-scale_factor} \quad (\text{D-20})$$

– If model_id is equal to 0, a frequency filtering model enables simulating the original film grain for c = 0..2, x = 0..PicWidthInSamples_L, and y = 0..PicHeightInSamples_L as specified by:

Depending on model_id, the synthesis of the film grain is modelled as follows.

- Otherwise (model_id is equal to 1), comp_model_value[c][i][j] shall be in the range of $-2^{(\text{filmGrainBitDepth}[c] - 1)}$ to $2^{(\text{filmGrainBitDepth}[c] - 1)}$ – 1, inclusive.
- If model_id is equal to 0, comp_model_value[c][i][j] shall be in the range of 0 to $2^{(\text{filmGrainBitDepth}[c] - 1)}$ – 1, inclusive.

elsewhere in this subclause.

comp_model_value[c][i][j] represents each one of the model values present for the colour component c and the intensity interval i. The set of model values has different meaning depending on the value of model_id. The value of comp_model_value[c][i][j] shall be constrained as follows, and may be additionally constrained as specified

- Otherwise (model_id is equal to 1), an auto-regression model enables simulating the original film grain for $c = 0..2$, $x = 0..PicWidthInSamples_L$, and $y = 0..PicHeightInSamples_L$ as specified by

$$\begin{aligned}
 G[x, y, c] = & (comp_model_value[c][s][0] * n[x, y, c] + \\
 & comp_model_value[c][s][1] * (G[x-1, y, c] + ((comp_model_value[c][s][4] * G[x, y-1, c]) >> \\
 & \log2_scale_factor)) + \\
 & comp_model_value[c][s][3] * ((comp_model_value[c][s][4] * G[x-1, y-1, c]) >> \\
 & \log2_scale_factor) + G[x+1, y-1, c]) + \\
 & comp_model_value[c][s][5] * (G[x-2, y, c] + \\
 & ((comp_model_value[c][s][4] * comp_model_value[c][s][4] * G[x, y-2, c]) >> \\
 & (2 * \log2_scale_factor))) + \\
 & comp_model_value[c][s][2] * G[x, y, c-1]) >> \log2_scale_factor
 \end{aligned} \tag{D-25}$$

where $n[x, y, c]$ is a random value with normalized Gaussian distribution (independent and identically distributed Gaussian random variable samples with zero mean and unity variance for each value of x , y , and c) and where the value of an element $G[x, y, c]$ used in the right-hand side of the equation is inferred to be equal to 0 when any of the following conditions are true:

- x is less than 0,
- y is less than 0,
- x is greater than or equal to $PicWidthInSamples_L$,
- c is less than 0.

comp_model_value[c][i][0] provides the first model value for the model as specified by model_id. **comp_model_value[c][i][0]** corresponds to the standard deviation of the Gaussian noise term in the generation functions specified in Equations D-20 through D-23.

comp_model_value[c][i][1] provides the second model value for the model as specified by model_id. **comp_model_value[c][i][1]** shall be greater than or equal to 0 and less than 16.

When not present in the film grain characteristics SEI message, **comp_model_value[c][i][1]** shall be inferred as follows.

- If model_id is equal to 0, **comp_model_value[c][i][1]** shall be inferred to be equal to 8.
- Otherwise (model_id is equal to 1), **comp_model_value[c][i][1]** shall be inferred to be equal to 0.

comp_model_value[c][i][1] is interpreted as follows.

- If model_id is equal to 0, **comp_model_value[c][i][1]** indicates the horizontal high cut frequency to be used to filter the DCT of a block of 16x16 random values.
- Otherwise (model_id is equal to 1), **comp_model_value[c][i][1]** indicates the first order spatial correlation for neighbouring samples $(x-1, y)$ and $(x, y-1)$.

comp_model_value[c][i][2] provides the third model value for the model as specified by model_id. **comp_model_value[c][i][2]** shall be greater than or equal to 0 and less than 16.

When not present in the film grain characteristics SEI message, **comp_model_value[c][i][2]** shall be inferred as follows.

- If model_id is equal to 0, **comp_model_value[c][i][2]** shall be inferred to be equal to **comp_model_value[c][i][1]**
- Otherwise (model_id is equal to 1), **comp_model_value[c][i][2]** shall be inferred to be equal to 0.

comp_model_value[c][i][2] is interpreted as follows.

- If model_id is equal to 0, **comp_model_value[c][i][2]** indicates the vertical high cut frequency to be used to filter the DCT of a block of 16x16 random values.
- Otherwise (model_id is equal to 1), **comp_model_value[c][i][2]** indicates the colour correlation between consecutive colour components.

comp_model_value[c][i][3] provides the fourth model value for the model as specified by model_id. **comp_model_value[c][i][3]** shall be greater than or equal to 0 and less than or equal to **comp_model_value[c][i][1]**.

When not present in the film grain characteristics SEI message, **comp_model_value[c][i][3]** shall be inferred to be equal to 0.

comp_model_value[c][i][3] is interpreted as follows.

- If model_id is equal to 0, comp_model_value[c][i][3] indicates the horizontal low cut frequency to be used to filter the DCT of a block of 16x16 random values.
- Otherwise (model_id is equal to 1), comp_model_value[c][i][3] indicates the first order spatial correlation for neighbouring samples (x-1, y-1) and (x+1, y-1).

comp_model_value[c][i][4] provides the fifth model value for the model as specified by model_id. comp_model_value[c][i][4] shall be greater than or equal to 0 and less than or equal to comp_model_value[c][i][2].

When not present in the film grain characteristics SEI message, comp_model_value[c][i][4] shall be inferred to be equal to model_id.

comp_model_value[c][i][4] is interpreted as follows.

- If model_id is equal to 0, comp_model_value[c][i][4] indicates the vertical low cut frequency to be used to filter the DCT of a block of 16x16 random values.
- Otherwise (model_id is equal to 1), comp_model_value[c][i][4] indicates the aspect ratio of the modelled grain.

comp_model_value[c][i][5] provides the sixth model value for the model as specified by model_id. When not present in the film grain characteristics SEI message, comp_model_value[c][i][5] shall be inferred to be equal to 0.

comp_model_value[c][i][5] is interpreted as follows.

- If model_id is equal to 0, comp_model_value[c][i][5] indicates the colour correlation between consecutive colour components.
- Otherwise (model_id is equal to 1), comp_model_value[c][i][5] indicates the second order spatial correlation for neighbouring samples (x, y-2) and (x-2, y).

film_grain_characteristics_repetition_period specifies the persistence of the film grain characteristics SEI message and may specify a picture order count interval within which another film grain characteristics SEI message or the end of the coded video sequence shall be present in the bitstream. The value of film_grain_characteristics_repetition_period shall be in the range 0 to 16 384, inclusive.

film_grain_characteristics_repetition_period equal to 0 specifies that the film grain characteristics SEI message applies to the current decoded picture only.

film_grain_characteristics_repetition_period equal to 1 specifies that the film grain characteristics SEI message persists in output order until any of the following conditions are true.

- A new coded video sequence begins, or
- A picture in an access unit containing a film grain characteristics SEI message that is output having PicOrderCnt() greater than PicOrderCnt(CurrPic).

film_grain_characteristics_repetition_period greater than 1 specifies that the film grain characteristics SEI message persists until any of the following conditions are true.

- A new coded video sequence begins, or
- A picture in an access unit containing a film grain characteristics SEI message is output having PicOrderCnt() greater than PicOrderCnt(CurrPic) and less than or equal to PicOrderCnt(CurrPic) + film_grain_characteristics_repetition_period.

film_grain_characteristics_repetition_period greater than 1 indicates that another film grain characteristics SEI message shall be present for a picture in an access unit that is output having PicOrderCnt() greater than PicOrderCnt(CurrPic) and less than or equal to PicOrderCnt(CurrPic) + film_grain_characteristics_repetition_period; unless the bitstream ends or a new coded video sequence begins without output of such a picture.

D.2.21 Deblocking filter display preference SEI message semantics

This SEI message provides the decoder with an indication of whether the display of the cropped result of the deblocking filter process specified in subclause 8.7 or of the cropped result of the picture construction process prior to the deblocking filter process specified in subclause 8.5.12 is preferred by the encoder for the display of each decoded picture that is output.

NOTE 1 – The display process is not specified in this Recommendation | International Standard. The means by which an encoder determines what to indicate as its preference expressed in a deblocking filter display preference SEI message is also not specified in this Recommendation | International Standard, and the expression of an expressed preference in a deblocking filter display preference SEI message does not impose any requirement on the display process.

deblocking_display_preference_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous deblocking filter display preference SEI message in output order. **deblocking_display_preference_cancel_flag** equal to 0 indicates that a **display_prior_to_deblocking_preferred_flag** and **deblocking_display_preference_repetition_period** follow.

NOTE 2 – In the absence of the deblocking filter display preference SEI message, or after the receipt of a deblocking filter display preference SEI message in which **deblocking_display_preference_cancel_flag** is equal to 1, the decoder should infer that the display of the cropped result of the deblocking filter process specified in subclause 8.7 is preferred over the display of the cropped result of the picture construction process prior to the deblocking filter process specified in subclause 8.5.12 for the display of each decoded picture that is output.

display_prior_to_deblocking_preferred_flag equal to 1 indicates that the encoder preference is for the display process (which is not specified in this Recommendation | International Standard) to display the cropped result of the picture construction process prior to the deblocking filter process specified in subclause 8.5.12 rather than the cropped result of the deblocking filter process specified in subclause 8.7 for each picture that is cropped and output as specified in Annex C. **display_prior_to_deblocking_preferred_flag** equal to 0 indicates that the encoder preference is for the display process (which is not specified in this Recommendation | International Standard) to display the cropped result of the deblocking filter process specified in subclause 8.7 rather than the cropped result of the picture construction process prior to the deblocking filter process specified in subclause 8.5.12 for each picture that is cropped and output as specified in Annex C.

NOTE 3 – The presence or absence of the deblocking filter display preference SEI message and the value of **display_prior_to_deblocking_preferred_flag** does not affect the requirements of the decoding process specified in this Recommendation | International Standard. Rather, it only provides an indication of when, in addition to fulfilling the requirements of this Recommendation | International Standard for the decoding process, enhanced visual quality might be obtained by performing the display process (which is not specified in this Recommendation | International Standard) in an alternative fashion. Encoders that use the deblocking filter display preference SEI message should be designed with an awareness that unless the encoder restricts its use of the DPB capacity specified in Annex A for the profile and level in use, some decoders may not have sufficient memory capacity for the storage of the result of the picture construction process prior to the deblocking filter process specified in subclause 8.5.12 in addition to the storage of the result of the deblocking filter process specified in subclause 8.7 when reordering and delaying pictures for display, and such decoders would therefore not be able to benefit from the preference indication. By restricting its use of the DPB capacity, an encoder can be able to use at least half of the DPB capacity specified in Annex A while allowing the decoder to use the remaining capacity for storage of unfiltered pictures that have been indicated as preferable for display until the output time arrives for those pictures.

dec_frame_buffering_constraint_flag equal to 1 indicates that the use of the frame buffering capacity of the HRD decoded picture buffer (DPB) as specified by **max_dec_frame_buffering** has been constrained such that the coded video sequence will not require a decoded picture buffer with more than $\text{Max}(1, \text{max_dec_frame_buffering})$ frame buffers to enable the output of the decoded filtered or unfiltered pictures, as indicated by the deblocking filter display preference SEI messages, at the output times specified by the **dpb_output_delay** of the picture timing SEI messages.

dec_frame_buffering_constraint_flag equal to 0 indicates that the use of the frame buffering capacity in the HRD may or may not be constrained in the manner that would be indicated by **dec_frame_buffering_constraint_flag** equal to 1.

For purposes of determining the constraint imposed when **dec_frame_buffering_constraint_flag** is equal to 1, the quantity of frame buffering capacity used at any given point in time by each frame buffer of the DPB that contains a picture shall be derived as follows:

- If both of the following criteria are satisfied for the frame buffer, the frame buffer is considered to use two frame buffers of capacity for its storage.
 - The frame buffer contains a frame or one or more fields that is marked as "used for reference", and
 - The frame buffer contains a picture for which both of the following criteria are fulfilled.
 - The HRD output time of the picture is greater than the given point in time, and
 - It has been indicated in a deblocking filter display preference SEI message that the encoder preference for the picture is for the display process to display the cropped result of the picture construction process prior to the deblocking filter process specified in subclause 8.5.12 rather than the cropped result of the deblocking filter process specified in subclause 8.7, and
- Otherwise, the frame buffer is considered to use one frame buffer of DPB capacity for its storage.

When **dec_frame_buffering_constraint_flag** is equal to 1, the frame buffering capacity used by all of the frame buffers in the DPB that contain pictures, as derived in this manner, shall not be greater than $\text{Max}(1, \text{max_dec_frame_buffering})$ during the operation of the HRD for the coded video sequence.

The value of **dec_frame_buffering_constraint_flag** shall be the same in all deblocking filter display preference SEI messages of the coded video sequence.

deblocking_display_preference_repetition_period specifies the persistence of the film grain characteristics SEI message and may specify a picture order count interval within which another film grain characteristics SEI message or the end of the coded video sequence shall be present in the bitstream. The value of **deblocking_display_preference_repetition_period** shall be in the range 0 to 16 384, inclusive.

deblocking_display_preference_repetition_period equal to 0 specifies that the deblocking filter display preference SEI message applies to the current decoded picture only.

deblocking_display_preference_repetition_period equal to 1 specifies that the deblocking filter display preference SEI message persists in output order until any of the following conditions are true.

- A new coded video sequence begins
- A picture in an access unit containing a deblocking filter display preference SEI message that is output having $\text{PicOrderCnt}()$ greater than $\text{PicOrderCnt}(\text{CurrPic})$.

deblocking_display_preference_repetition_period greater than 1 specifies that the deblocking filter display preference SEI message persists until any of the following conditions are true.

- A new coded video sequence begins
- A picture in an access unit containing a deblocking filter display preference SEI message is output having $\text{PicOrderCnt}()$ greater than $\text{PicOrderCnt}(\text{CurrPic})$ and less than or equal to $\text{PicOrderCnt}(\text{CurrPic}) + \text{deblocking_display_preference_repetition_period}$.

deblocking_display_preference_repetition_period greater than 1 indicates that another deblocking filter display preference SEI message shall be present for a picture in an access unit that is output having $\text{PicOrderCnt}()$ greater than $\text{PicOrderCnt}(\text{CurrPic})$ and less than or equal to $\text{PicOrderCnt}(\text{CurrPic}) + \text{deblocking_display_preference_repetition_period}$; unless the bitstream ends or a new coded video sequence begins without output of such a picture.

D.2.22 Stereo video information SEI message semantics

This SEI message provides the decoder with an indication that the entire coded video sequence consists of pairs of pictures forming stereo-view content.

The stereo video information SEI message shall not be present in any access unit of a coded video sequence unless a stereo video information SEI message is present in the first access unit of the coded video sequence.

field_views_flag equal to 1 indicates that all pictures in the current coded video sequence are fields and all fields of a particular parity are considered a left view and all fields of the opposite parity are considered a right view for stereo-view content. **field_views_flag** equal to 0 indicates that all pictures in the current coded video sequence are frames and alternating frames in output order represent a view of a stereo view. The value of **field_views_flag** shall be the same in all stereo video information SEI messages within a coded video sequence.

When the stereo video information SEI message is present and **field_views_flag** is equal to 1, the left view and right view of a stereo video pair shall be coded as a complementary field pair, the display time of the first field of the field pair in output order should be delayed to coincide with the display time of the second field of the field pair in output order, and the spatial locations of the samples in each individual field should be interpreted for display purposes as representing complete pictures as shown in Figure 6-1 rather than as spatially-distinct fields within a frame as shown in Figure 6-2.

NOTE – The display process is not specified in this Recommendation | International Standard.

top_field_is_left_view_flag equal to 1 indicates that the top fields in the coded video sequence represent a left view and the bottom fields in the coded video sequence represent a right view. **top_field_is_left_view_flag** equal to 0 indicates that the bottom fields in the coded video sequence represent a left view and the top fields in the coded video sequence represent a right view. When present, the value of **top_field_is_left_view_flag** shall be the same in all stereo video information SEI messages within a coded video sequence.

current_frame_is_left_view_flag equal to 1 indicates that the current picture is the left view of a stereo-view pair. **current_frame_is_left_view_flag** equal to 0 indicates that the current picture is the right view of a stereo-view pair.

next_frame_is_second_view_flag equal to 1 indicates that the current picture and the next picture in output order form a stereo-view pair, and the display time of the current picture should be delayed to coincide with the display time of the next picture in output order. **next_frame_is_second_view_flag** equal to 0 indicates that the current picture and the previous picture in output order form a stereo-view pair, and the display time of the current picture should not be delayed for purposes of stereo-view pairing.

left_view_self_contained_flag equal to 1 indicates that no inter prediction operations within the decoding process for the left-view pictures of the coded video sequence refer to reference pictures that are right-view pictures. **left_view_self_contained_flag** equal to 0 indicates that some inter prediction operations within the decoding process for the left-view pictures of the coded video sequence may or may not refer to reference pictures that are right-view pictures. Within a coded video sequence, the value of **left_view_self_contained_flag** in all stereo video information SEI messages shall be the same.

right_view_self_contained_flag equal to 1 indicates that no inter prediction operations within the decoding process for the right-view pictures of the coded video sequence refer to reference pictures that are left-view pictures. **right_view_self_contained_flag** equal to 0 indicates that some inter prediction operations within the decoding process for the right-view pictures of the coded video sequence may or may not refer to reference pictures that are left-view pictures. Within a coded video sequence, the value of **right_view_self_contained_flag** in all stereo video information SEI messages shall be the same.

D.2.23 Reserved SEI message semantics

This message consists of data reserved for future backward-compatible use by ITU-T | ISO/IEC. Encoders conforming to this Recommendation | International Standard shall not send reserved SEI messages until and unless the use of such messages has been specified by ITU-T | ISO/IEC. Decoders conforming to this Recommendation | International Standard that encounter reserved SEI messages shall discard their content without effect on the decoding process, except as specified in future Recommendations | International Standards specified by ITU-T | ISO/IEC.

reserved_sei_message_payload_byte is a byte reserved for future use by ITU-T | ISO/IEC.

Annex E

Video usability information

(This annex forms an integral part of this Recommendation | International Standard)

This Annex specifies syntax and semantics of the VUI parameters of the sequence parameter sets.

VUI parameters are not required for constructing the luma or chroma samples by the decoding process. Conforming decoders are not required to process this information for output order conformance to this Recommendation | International Standard (see Annex C for the specification of conformance). Some VUI parameters are required to check bitstream conformance and for output timing decoder conformance.

In Annex E, specification for presence of VUI parameters is also satisfied when those parameters (or some subset of them) are conveyed to decoders (or to the HRD) by other means not specified by this Recommendation | International Standard. When present in the bitstream, VUI parameters shall follow the syntax and semantics specified in subclauses 7.3.2.1 and 7.4.2.1 and this annex. When the content of VUI parameters is conveyed for the application by some means other than presence within the bitstream, the representation of the content of the VUI parameters is not required to use the same syntax specified in this annex. For the purpose of counting bits, only the appropriate bits that are actually present in the bitstream are counted.

E.1 VUI syntax

E.1.1 VUI parameters syntax

vui_parameters() {	C	Descriptor
aspect_ratio_info_present_flag	0	u(1)
if(aspect_ratio_info_present_flag) {		
aspect_ratio_idc	0	u(8)
if(aspect_ratio_idc == Extended_SAR) {		
sar_width	0	u(16)
sar_height	0	u(16)
}		
}		
overscan_info_present_flag	0	u(1)
if(overscan_info_present_flag)		
overscan_appropriate_flag	0	u(1)
video_signal_type_present_flag	0	u(1)
if(video_signal_type_present_flag) {		
video_format	0	u(3)
video_full_range_flag	0	u(1)
colour_description_present_flag	0	u(1)
if(colour_description_present_flag) {		
colour_primaries	0	u(8)
transfer_characteristics	0	u(8)
matrix_coefficients	0	u(8)
}		
}		
chroma_loc_info_present_flag	0	u(1)
if(chroma_loc_info_present_flag) {		
chroma_sample_loc_type_top_field	0	ue(v)
chroma_sample_loc_type_bottom_field	0	ue(v)
}		
timing_info_present_flag	0	u(1)
if(timing_info_present_flag) {		
num_units_in_tick	0	u(32)
time_scale	0	u(32)
fixed_frame_rate_flag	0	u(1)
}		
nal_hrd_parameters_present_flag	0	u(1)
if(nal_hrd_parameters_present_flag)		
hrd_parameters()		
vcl_hrd_parameters_present_flag	0	u(1)
if(vcl_hrd_parameters_present_flag)		
hrd_parameters()		
if(nal_hrd_parameters_present_flag vcl_hrd_parameters_present_flag)		
low_delay_hrd_flag	0	u(1)
pic_struct_present_flag	0	u(1)
bitstream_restriction_flag	0	u(1)
if(bitstream_restriction_flag) {		

motion_vectors_over_pic_boundaries_flag	0	u(1)
max_bytes_per_pic_denom	0	ue(v)
max_bits_per_mb_denom	0	ue(v)
log2_max_mv_length_horizontal	0	ue(v)
log2_max_mv_length_vertical	0	ue(v)
num_reorder_frames	0	ue(v)
max_dec_frame_buffering	0	ue(v)
}		
}		

E.1.2 HRD parameters syntax

hrd_parameters() {	C	Descriptor
cpb_cnt_minus1	0	ue(v)
bit_rate_scale	0	u(4)
cpb_size_scale	0	u(4)
for(SchedSelIdx = 0; SchedSelIdx <= cpb_cnt_minus1; SchedSelIdx++) {		
bit_rate_value_minus1[SchedSelIdx]	0	ue(v)
cpb_size_value_minus1[SchedSelIdx]	0	ue(v)
cbr_flag[SchedSelIdx]	0	u(1)
}		
initial_cpb_removal_delay_length_minus1	0	u(5)
cpb_removal_delay_length_minus1	0	u(5)
dpb_output_delay_length_minus1	0	u(5)
time_offset_length	0	u(5)
}		

E.2 VUI semantics

E.2.1 VUI parameters semantics

aspect_ratio_info_present_flag equal to 1 specifies that **aspect_ratio_idc** is present. **aspect_ratio_info_present_flag** equal to 0 specifies that **aspect_ratio_idc** is not present.

aspect_ratio_idc specifies the value of the sample aspect ratio of the luma samples. Table E-1 shows the meaning of the code. When **aspect_ratio_idc** indicates Extended_SAR, the sample aspect ratio is represented by **sar_width** and **sar_height**. When the **aspect_ratio_idc** syntax element is not present, **aspect_ratio_idc** value shall be inferred to be equal to 0.

Table E-1 – Meaning of sample aspect ratio indicator

aspect_ratio_idc	Sample aspect ratio	(informative) Examples of use
0	Unspecified	
1	1:1 ("square")	1280x720 16:9 frame without horizontal overscan 1920x1080 16:9 frame without horizontal overscan (cropped from 1920x1088) 640x480 4:3 frame without horizontal overscan
2	12:11	720x576 4:3 frame with horizontal overscan 352x288 4:3 frame without horizontal overscan
3	10:11	720x480 4:3 frame with horizontal overscan 352x240 4:3 frame without horizontal overscan
4	16:11	720x576 16:9 frame with horizontal overscan 528x576 4:3 frame without horizontal overscan
5	40:33	720x480 16:9 frame with horizontal overscan 528x480 4:3 frame without horizontal overscan
6	24:11	352x576 4:3 frame without horizontal overscan 480x576 16:9 frame with horizontal overscan
7	20:11	352x480 4:3 frame without horizontal overscan 480x480 16:9 frame with horizontal overscan
8	32:11	352x576 16:9 frame without horizontal overscan
9	80:33	352x480 16:9 frame without horizontal overscan
10	18:11	480x576 4:3 frame with horizontal overscan
11	15:11	480x480 4:3 frame with horizontal overscan
12	64:33	528x576 16:9 frame without horizontal overscan
13	160:99	528x480 16:9 frame without horizontal overscan
14	4:3	1440x1080 16:9 frame without horizontal overscan
15	3:2	1280x1080 16:9 frame without horizontal overscan
16	2:1	960x1080 16:9 frame without horizontal overscan
17..254	Reserved	
255	Extended_SAR	

sar_width indicates the horizontal size of the sample aspect ratio (in arbitrary units).

sar_height indicates the vertical size of the sample aspect ratio (in the same arbitrary units as sar_width).

sar_width and sar_height shall be relatively prime or equal to 0. When aspect_ratio_idc is equal to 0 or sar_width is equal to 0 or sar_height is equal to 0, the sample aspect ratio shall be considered unspecified by this Recommendation | International Standard.

overscan_info_present_flag equal to 1 specifies that the overscan_appropriate_flag is present. When overscan_info_present_flag is equal to 0 or is not present, the preferred display method for the video signal is unspecified.

overscan_appropriate_flag equal to 1 indicates that the cropped decoded pictures output are suitable for display using overscan. overscan_appropriate_flag equal to 0 indicates that the cropped decoded pictures output contain visually important information in the entire region out to the edges of the cropping rectangle of the picture, such that the cropped decoded pictures output should not be displayed using overscan. Instead, they should be displayed using either an exact match between the display area and the cropping rectangle, or using underscan.

NOTE 1 – For example, overscan_appropriate_flag equal to 1 might be used for entertainment television programming, or for a live view of people in a videoconference, and overscan_appropriate_flag equal to 0 might be used for computer screen capture or security camera content.

video_signal_type_present_flag equal to 1 specifies that video_format, video_full_range_flag and colour_description_present_flag are present. video_signal_type_present_flag equal to 0, specify that video_format, video_full_range_flag and colour_description_present_flag are not present.

video_format indicates the representation of the pictures as specified in Table E-2, before being coded in accordance with this Recommendation | International Standard. When the video_format syntax element is not present, video_format value shall be inferred to be equal to 5.

Table E-2 – Meaning of video_format

video_format	Meaning
0	Component
1	PAL
2	NTSC
3	SECAM
4	MAC
5	Unspecified video format
6	Reserved
7	Reserved

video_full_range_flag indicates the black level and range of the luma and chroma signals as derived from E'_Y , E'_{PB} , and E'_{PR} or E'_R , E'_G , and E'_B analogue component signals.

When the video_full_range_flag syntax element is not present, the value of video_full_range_flag shall be inferred to be equal to 0.

colour_description_present_flag equal to 1 specifies that colour_primaries, transfer_characteristics and matrix_coefficients are present. colour_description_present_flag equal to 0 specifies that colour_primaries, transfer_characteristics and matrix_coefficients are not present.

colour_primaries indicates the chromaticity coordinates of the source primaries as specified in Table E-3 in terms of the CIE 1931 definition of x and y as specified by ISO/CIE 10527.

When the colour_primaries syntax element is not present, the value of colour_primaries shall be inferred to be equal to 2 (the chromaticity is unspecified or is determined by the application).

Table E-3 – Colour primaries

Value	Primaries			Informative Remark
0	Reserved			For future use by ITU-T / ISO/IEC
1	primary	x	y	ITU-R Recommendation BT.709-5
	green	0.300	0.600	
	blue	0.150	0.060	
	red	0.640	0.330	
	white D65	0.3127	0.3290	
2	Unspecified			Image characteristics are unknown or are determined by the application.
3	Reserved			
4	primary	x	y	ITU-R Recommendation BT.470-6 System M
	green	0.21	0.71	
	blue	0.14	0.08	
	red	0.67	0.33	
	white C	0.310	0.316	
5	primary	x	y	ITU-R Recommendation BT.470-6 System B, G
	green	0.29	0.60	
	blue	0.15	0.06	
	red	0.64	0.33	
	white D65	0.3127	0.3290	
6	primary	x	y	Society of Motion Picture and Television Engineers 170M (1999)
	green	0.310	0.595	
	blue	0.155	0.070	
	red	0.630	0.340	
	white D65	0.3127	0.3290	
7	primary	x	y	Society of Motion Picture and Television Engineers 240M (1999)
	green	0.310	0.595	
	blue	0.155	0.070	
	red	0.630	0.340	
	white D65	0.3127	0.3290	
8	primary	x	y	Generic film (colour filters using Illuminant C)
	green	0.243	0.692 (Wratten 58)	
	blue	0.145	0.049 (Wratten 47)	
	red	0.681	0.319 (Wratten 25)	
	white C	0.310	0.316	
9-255	Reserved			For future use by ITU-T / ISO/IEC

transfer_characteristics indicates the opto-electronic transfer characteristic of the source picture as specified in Table E-4 as a function of a linear optical intensity input L_c with an analogue range of 0 to 1.

When the transfer_characteristics syntax element is not present, the value of transfer_characteristics shall be inferred to be equal to 2 (the transfer characteristics are unspecified or are determined by the application).

Table E-4 – Transfer characteristics

Value	Transfer Characteristic	Informative Remark
0	Reserved	For future use by ITU-T / ISO/IEC
1	$V = 1.099 L_c^{0.45} - 0.099$ for $1 \geq L_c \geq 0.018$ $V = 4.500 L_c$ for $0.018 > L_c \geq 0$	ITU-R Recommendation BT.709-5
2	Unspecified	Image characteristics are unknown or are determined by the application.
3	Reserved	For future use by ITU-T / ISO/IEC
4	Assumed display gamma 2.2	ITU-R Recommendation BT.470-6 System M
5	Assumed display gamma 2.8	ITU-R Recommendation BT.470-6 System B, G
6	$V = 1.099 L_c^{0.45} - 0.099$ for $1 \geq L_c \geq 0.018$ $V = 4.500 L_c$ for $0.018 > L_c \geq 0$	Society of Motion Picture and Television Engineers 170M (1999)
7	$V = 1.1115 L_c^{0.45} - 0.1115$ for $1 \geq L_c \geq 0.0228$ $V = 4.0 L_c$ for $0.0228 > L_c \geq 0$	Society of Motion Picture and Television Engineers 240M (1999)
8	$V = L_c$ for $1 > L_c \geq 0$	Linear transfer characteristics
9	$V = 1.0 - \text{Log}_{10}(L_c) \div 2$ for $1 \geq L_c \geq 0.01$ $V = 0.0$ for $0.01 > L_c \geq 0$	Logarithmic transfer characteristic (100:1 range)
10	$V = 1.0 - \text{Log}_{10}(L_c) \div 2.5$ for $1 \geq L_c \geq 0.0031622777$ $V = 0.0$ for $0.0031622777 > L_c \geq 0$	Logarithmic transfer characteristic (316.22777:1 range)
11..255	Reserved	For future use by ITU-T / ISO/IEC

matrix_coefficients describes the matrix coefficients used in deriving luma and chroma signals from the green, blue, and red primaries, as specified in Table E-5.

matrix_coefficients shall not be equal to 0 unless both of the following conditions are true

- BitDepth_C is equal to BitDepth_Y
- chroma_format_idc is equal to 3 (4:4:4)

The specification of the use of matrix_coefficients equal to 0 under all other conditions is reserved for future use by ITU-T | ISO/IEC.

matrix_coefficients shall not be equal to 8 unless one or both of the following conditions are true

- BitDepth_C is equal to BitDepth_Y
- BitDepth_C is equal to BitDepth_Y + 1 and chroma_format_idc is equal to 3 (4:4:4)

The specification of the use of matrix_coefficients equal to 8 under all other conditions is reserved for future use by ITU-T | ISO/IEC.

When the matrix_coefficients syntax element is not present, the value of matrix_coefficients shall be inferred to be equal to 2.

The interpretation of matrix_coefficients is defined as follows.

- E'_R, E'_G, and E'_B are analogue with values in the range of 0 to 1.
- White is specified as having E'_R equal to 1, E'_G equal to 1, and E'_B equal to 1.
- Black is specified as having E'_R equal to 0, E'_G equal to 0, and E'_B equal to 0.
- If video_full_range_flag is equal to 0, the following equations apply.
 - If matrix_coefficients is equal to 1, 4, 5, 6, or 7, the following equations apply.

$$Y = \text{Round}((1 \ll (\text{BitDepth}_Y - 8)) * (219 * E'_Y + 16)) \quad (\text{E-1})$$

$$Cb = \text{Round}((1 \ll (\text{BitDepth}_C - 8)) * (224 * E'_{PB} + 128)) \quad (\text{E-2})$$

$$Cr = \text{Round}((1 \ll (\text{BitDepth}_C - 8)) * (224 * E'_{PR} + 128)) \quad (\text{E-3})$$

- Otherwise, if matrix_coefficients is equal to 0 or 8, the following equations apply.

$$R = (1 \ll (\text{BitDepth}_Y - 8)) * (219 * E'_R + 16) \quad (\text{E-4})$$

$$G = (1 \ll (\text{BitDepth}_Y - 8)) * (219 * E'_G + 16) \quad (\text{E-5})$$

$$B = (1 \ll (\text{BitDepth}_Y - 8)) * (219 * E'_B + 16) \quad (\text{E-6})$$

- Otherwise, if matrix_coefficients is equal to 2, the interpretation of the matrix_coefficients syntax element is unknown or is determined by the application.
- Otherwise (matrix_coefficients is not equal to 0, 1, 2, 4, 5, 6, 7, or 8), the interpretation of the matrix_coefficients syntax element is reserved for future definition by ITU-T | ISO/IEC.
- Otherwise (video_full_range_flag is equal to 1), the following equations apply.
 - If matrix_coefficients is equal to 1, 4, 5, 6, or 7, the following equations apply.

$$Y = \text{Round}((1 \ll \text{BitDepth}_Y) - 1) * E'_Y \quad (\text{E-7})$$

$$Cb = \text{Round}(((1 \ll \text{BitDepth}_C) - 1) * E'_{PB} + (1 \ll (\text{BitDepth}_C - 1))) \quad (\text{E-8})$$

$$Cr = \text{Round}(((1 \ll \text{BitDepth}_C) - 1) * E'_{PR} + (1 \ll (\text{BitDepth}_C - 1))) \quad (\text{E-9})$$

- Otherwise, if matrix_coefficients is equal to 0 or 8, the following equations apply.

$$R = ((1 \ll \text{BitDepth}_Y) - 1) * E'_R \quad (\text{E-10})$$

$$G = ((1 \ll \text{BitDepth}_Y) - 1) * E'_G \quad (\text{E-11})$$

$$B = ((1 \ll \text{BitDepth}_Y) - 1) * E'_B \quad (\text{E-12})$$

- Otherwise, if matrix_coefficients is equal to 2, the interpretation of the matrix_coefficients syntax element is unknown or is determined by the application.
- Otherwise (matrix_coefficients is not equal to 0, 1, 2, 4, 5, 6, 7, or 8), the interpretation of the matrix_coefficients syntax element is reserved for future definition by ITU-T | ISO/IEC.
- If matrix_coefficients is not equal to 0 or 8, the following equations apply.

$$E'_Y = K_R * E'_R + (1 - K_R - K_B) * E'_G + K_B * E'_B \quad (\text{E-13})$$

$$E'_{PB} = 0.5 * (E'_B - E'_Y) \div (1 - K_B) \quad (\text{E-14})$$

$$E'_{PR} = 0.5 * (E'_R - E'_Y) \div (1 - K_R) \quad (\text{E-15})$$

NOTE 2 – Then E'_Y is analogue with values in the range of 0 to 1, E'_{PB} and E'_{PR} are analogue with values in the range of -0.5 to 0.5, and white is equivalently given by $E'_Y = 1$, $E'_{PB} = 0$, $E'_{PR} = 0$.

- Otherwise, if matrix_coefficients is equal to 0, the following equations apply.

$$Y = \text{Round}(G) \quad (\text{E-16})$$

$$Cb = \text{Round}(B) \quad (\text{E-17})$$

$$Cr = \text{Round}(R) \quad (\text{E-18})$$

- Otherwise (matrix_coefficients is equal to 8), the following applies.
 - If BitDepth_C is equal to BitDepth_Y, the following equations apply.

$$Y = \text{Round}(0.5 * G + 0.25 * (R + B)) \quad (\text{E-19})$$

$$Cb = \text{Round}(0.5 * G - 0.25 * (R + B)) \quad (\text{E-20})$$

$$Cr = \text{Round}(0.5 * (R - B)) \quad (\text{E-21})$$

NOTE 3 – For purposes of the YCgCo nomenclature used in Table E-5, Cb and Cr of Equations E-20 and E-21 may be referred to as Cg and Co, respectively. The inverse conversion for the above four equations should be computed as.

$$t = Y - Cb \quad (\text{E-22})$$

$$G = Y + Cb \quad (\text{E-23})$$

$$B = t - Cr \quad (\text{E-24})$$

$$R = t + Cr \quad (\text{E-25})$$

- Otherwise (BitDepth_C is not equal to BitDepth_Y), the following equations apply.

$$Cr = \text{Round}(R) - \text{Round}(B) \quad (\text{E-26})$$

$$t = \text{Round}(B) + (Cr \gg 1) \quad (\text{E-27})$$

$$Cb = \text{Round}(G) - t \quad (\text{E-28})$$

$$Y = t + (Cb \gg 1) \quad (\text{E-29})$$

NOTE 4 – For purposes of the YCgCo nomenclature used in Table E-5, Cb and Cr of Equations E-28 and E-26 may be referred to as Cg and Co, respectively. The inverse conversion for the above four equations should be computed as.

$$t = Y - (Cb \gg 1) \quad (\text{E-30})$$

$$G = t + Cb \quad (\text{E-31})$$

$$B = t - (Cr \gg 1) \quad (\text{E-32})$$

$$R = B + Cr \quad (\text{E-33})$$

Table E-5 – Matrix coefficients

Value	Matrix	Informative remark
0	GBR	Typically referred to as RGB; see Equations E-16 to E-18
1	$K_R = 0.2126$; $K_B = 0.0722$	ITU-R Recommendation BT.709-5 and Society of Motion Picture and Television Engineers RP177 (1993)
2	Unspecified	Image characteristics are unknown or are determined by the application.
3	Reserved	For future use by ITU-T / ISO/IEC
4	$K_R = 0.30$; $K_B = 0.11$	United States Federal Communications Commission Title 47 Code of Federal Regulations (2003) 73.682 (a) (20)
5	$K_R = 0.299$; $K_B = 0.114$	ITU-R Recommendation BT.470-6 System B, G
6	$K_R = 0.299$; $K_B = 0.114$	Society of Motion Picture and Television Engineers 170M (1999)
7	$K_R = 0.212$; $K_B = 0.087$	Society of Motion Picture and Television Engineers 240M (1999)
8	YCgCo	See Equations E-19 to E-33
9-255	Reserved	For future use by ITU-T / ISO/IEC

chroma_loc_info_present_flag equal to 1 specifies that **chroma_sample_loc_type_top_field** and **chroma_sample_loc_type_bottom_field** are present. **chroma_loc_info_present_flag** equal to 0 specifies that **chroma_sample_loc_type_top_field** and **chroma_sample_loc_type_bottom_field** are not present.

chroma_sample_loc_type_top_field and **chroma_sample_loc_type_bottom_field** specify the location of chroma samples for the top field and the bottom field as shown in Figure E-1. The value of **chroma_sample_loc_type_top_field** and **chroma_sample_loc_type_bottom_field** shall be in the range of 0 to 5, inclusive. When the **chroma_sample_loc_type_top_field** and **chroma_sample_loc_type_bottom_field** are not present, the values of **chroma_sample_loc_type_top_field** and **chroma_sample_loc_type_bottom_field** shall be inferred to be equal to 0.

NOTE 5 – When coding progressive source material, **chroma_sample_loc_type_top_field** and **chroma_sample_loc_type_bottom_field** should have the same value.

For each picture n where n indicates the n -th picture (in output order) that is output and picture n is not the last picture in the bitstream (in output order) that is output, the value of $\Delta t_{fi,dpb}(n)$ is specified by

$$\Delta t_{fi,dpb}(n) = \Delta t_{o,dpb}(n) \div \text{DeltaTfiDivisor} \quad (\text{E-34})$$

where $\Delta t_{o,dpb}(n)$ is specified in Equation C-13 and DeltaTfiDivisor is specified by Table E-6 based on the value of pic_struct_present_flag, field_pic_flag, and pic_struct for the coded video sequence containing picture n . Entries marked "-" in Table E-6 indicate a lack of dependence of DeltaTfiDivisor on the corresponding syntax element.

When fixed_frame_rate_flag is equal to 1 for a coded video sequence containing picture n , the value computed for $\Delta t_{fi,dpb}(n)$ shall be equal to t_c as specified in Equation C-1 (using the value of t_c for the coded video sequence containing picture n) when either or both of the following conditions are true for the following picture n_n that is specified for use in Equation C-13.

- picture n_n is in the same coded video sequence as picture n .
- picture n_n is in a different coded video sequence and fixed_frame_rate_flag is equal to 1 in the coded video sequence containing picture n_n and the value of num_units_in_tick \div time_scale is the same for both coded video sequences.

Table E-6 – Divisor for computation of $\Delta t_{fi,dpb}(n)$

pic_struct_present_flag	field_pic_flag	pic_struct	DeltaTfiDivisor
0	1	-	1
1	-	1	1
1	-	2	1
0	0	-	2
1	-	0	2
1	-	3	2
1	-	4	2
1	-	5	3
1	-	6	3
1	-	7	4
1	-	8	6

nal_hrd_parameters_present_flag equal to 1 specifies that NAL HRD parameters (pertaining to Type II bitstream conformance) are present. nal_hrd_parameters_present_flag equal to 0 specifies that NAL HRD parameters are not present.

NOTE 6 – When nal_hrd_parameters_present_flag is equal to 0, the conformance of the bitstream cannot be verified without provision of the NAL HRD parameters, including the NAL sequence HRD parameter information and all buffering period and picture timing SEI messages, by some means not specified in this Recommendation | International Standard.

When nal_hrd_parameters_present_flag is equal to 1, NAL HRD parameters (subclauses E.1.2 and E.2.2) immediately follow the flag.

The variable NalHrdBpPresentFlag is derived as follows.

- If any of the following is true, the value of NalHrdBpPresentFlag shall be set equal to 1.
 - nal_hrd_parameters_present_flag is present in the bitstream and is equal to 1
 - the need for presence of buffering periods for NAL HRD operation to be present in the bitstream in buffering period SEI messages is determined by the application, by some means not specified in this Recommendation | International Standard.
- Otherwise, the value of NalHrdBpPresentFlag shall be set equal to 0.

vcl_hrd_parameters_present_flag equal to 1 specifies that VCL HRD parameters (pertaining to all bitstream conformance) are present. vcl_hrd_parameters_present_flag equal to 0 specifies that VCL HRD parameters are not present.

NOTE 7 – When vcl_hrd_parameters_present_flag is equal to 0, the conformance of the bitstream cannot be verified without provision of the VCL HRD parameters and all buffering period and picture timing SEI messages, by some means not specified in this Recommendation | International Standard.

When `vcl_hrd_parameters_present_flag` is equal to 1, VCL HRD parameters (subclauses E.1.2 and E.2.2) immediately follow the flag.

The variable `VclHrdBpPresentFlag` is derived as follows.

- If any of the following is true, the value of `VclHrdBpPresentFlag` shall be set equal to 1.
 - `vcl_hrd_parameters_present_flag` is present in the bitstream and is equal to 1
 - the need for presence of buffering periods for VCL HRD operation to be present in the bitstream in buffering period SEI messages is determined by the application, by some means not specified in this Recommendation | International Standard.
- Otherwise, the value of `VclHrdBpPresentFlag` shall be set equal to 0.

The variable `CpbDpbDelaysPresentFlag` is derived as follows.

- If any of the following is true, the value of `CpbDpbDelaysPresentFlag` shall be set equal to 1.
 - `nal_hrd_parameters_present_flag` is present in the bitstream and is equal to 1
 - `vcl_hrd_parameters_present_flag` is present in the bitstream and is equal to 1
 - the need for presence of CPB and DPB output delays to be present in the bitstream in picture timing SEI messages is determined by the application, by some means not specified in this Recommendation | International Standard.
- Otherwise, the value of `CpbDpbDelaysPresentFlag` shall be set equal to 0.

low_delay_hrd_flag specifies the HRD operational mode as specified in Annex C. When `fixed_frame_rate_flag` is equal to 1, `low_delay_hrd_flag` shall be equal to 0.

NOTE 8 – When `low_delay_hrd_flag` is equal to 1, "big pictures" that violate the nominal CPB removal times due to the number of bits used by an access unit are permitted. It is expected, but not required, that such "big pictures" occur only occasionally.

pic_struct_present_flag equal to 1 specifies that picture timing SEI messages (subclause D.2.2) are present that include the `pic_struct` syntax element. `pic_struct_present_flag` equal to 0 specifies that the `pic_struct` syntax element is not present in picture timing SEI messages. When `pic_struct_present_flag` is not present, its value shall be inferred to be equal to 0.

bitstream_restriction_flag equal to 1, specifies that the following coded video sequence bitstream restriction parameters are present. `bitstream_restriction_flag` equal to 0, specifies that the following coded video sequence bitstream restriction parameters are not present.

motion_vectors_over_pic_boundaries_flag equal to 0 indicates that no sample outside the picture boundaries and no sample at a fractional sample position whose value is derived using one or more samples outside the picture boundaries is used to inter predict any sample. `motion_vectors_over_pic_boundaries_flag` equal to 1 indicates that one or more samples outside picture boundaries may be used in inter prediction. When the `motion_vectors_over_pic_boundaries_flag` syntax element is not present, `motion_vectors_over_pic_boundaries_flag` value shall be inferred to be equal to 1.

max_bytes_per_pic_denom indicates a number of bytes not exceeded by the sum of the sizes of the VCL NAL units associated with any coded picture in the coded video sequence.

The number of bytes that represent a picture in the NAL unit stream is specified for this purpose as the total number of bytes of VCL NAL unit data (i.e., the total of the `NumBytesInNALunit` variables for the VCL NAL units) for the picture. The value of `max_bytes_per_pic_denom` shall be in the range of 0 to 16, inclusive.

Depending on `max_bytes_per_pic_denom` the following applies.

- If `max_bytes_per_pic_denom` is equal to 0, no limits are indicated.
- Otherwise (`max_bytes_per_pic_denom` is not equal to 0), no coded picture shall be represented in the coded video sequence by more than the following number of bytes.

$$(\text{PicSizeInMbs} * \text{RawMbBits}) \div (8 * \text{max_bytes_per_pic_denom}) \quad (\text{E-35})$$

When the `max_bytes_per_pic_denom` syntax element is not present, the value of `max_bytes_per_pic_denom` shall be inferred to be equal to 2.

max_bits_per_mb_denom indicates the maximum number of coded bits of `macroblock_layer()` data for any macroblock in any picture of the coded video sequence. The value of `max_bits_per_mb_denom` shall be in the range of 0 to 16, inclusive.

Depending on `max_bits_per_mb_denom` the following applies.

- If `max_bits_per_mb_denom` is equal to 0, no limit is specified.
- Otherwise (`max_bits_per_mb_denom` is not equal to 0), no coded `macroblock_layer()` shall be represented in the bitstream by more than the following number of bits.

$$(128 + \text{RawMbBits}) \div \text{max_bits_per_mb_denom} \quad (\text{E-36})$$

Depending on `entropy_coding_mode_flag`, the bits of `macroblock_layer()` data are counted as follows.

- If `entropy_coding_mode_flag` is equal to 0, the number of bits of `macroblock_layer()` data is given by the number of bits in the `macroblock_layer()` syntax structure for a macroblock.
- Otherwise (`entropy_coding_mode_flag` is equal to 1), the number of bits of `macroblock_layer()` data for a macroblock is given by the number of times `read_bits(1)` is called in subclauses 9.3.3.2.2 and 9.3.3.2.3 when parsing the `macroblock_layer()` associated with the macroblock.

When the `max_bits_per_mb_denom` is not present, the value of `max_bits_per_mb_denom` shall be inferred to be equal to 1.

`log2_max_mv_length_horizontal` and **`log2_max_mv_length_vertical`** indicate the maximum absolute value of a decoded horizontal and vertical motion vector component, respectively, in $\frac{1}{4}$ luma sample units, for all pictures in the coded video sequence. A value of n asserts that no value of a motion vector component shall exceed the range from -2^n to $2^n - 1$, inclusive, in units of $\frac{1}{4}$ luma sample displacement. The value of `log2_max_mv_length_horizontal` shall be in the range of 0 to 16, inclusive. The value of `log2_max_mv_length_vertical` shall be in the range of 0 to 16, inclusive. When `log2_max_mv_length_horizontal` is not present, the values of `log2_max_mv_length_horizontal` and `log2_max_mv_length_vertical` shall be inferred to be equal to 16.

NOTE 9 – The maximum absolute value of a decoded vertical or horizontal motion vector component is also constrained by profile and level limits as specified in Annex A.

`num_reorder_frames` indicates the maximum number of frames, complementary field pairs, or non-paired fields that precede any frame, complementary field pair, or non-paired field in the coded video sequence in decoding order and follow it in output order. The value of `num_reorder_frames` shall be in the range of 0 to `max_dec_frame_buffering`, inclusive. When the `num_reorder_frames` syntax element is not present, the value of `num_reorder_frames` value shall be inferred to be equal to `max_dec_frame_buffering`.

`max_dec_frame_buffering` specifies the required size of the HRD decoded picture buffer (DPB) in units of frame buffers. The coded video sequence shall not require a decoded picture buffer with size of more than $\text{Max}(1, \text{max_dec_frame_buffering})$ frame buffers to enable the output of decoded pictures at the output times specified by `dpb_output_delay` of the picture timing SEI messages. The value of `max_dec_frame_buffering` shall be in the range of `num_ref_frames` to `MaxDpbSize` (as specified in subclause A.3.1 or A.3.2), inclusive. When the `max_dec_frame_buffering` syntax element is not present, the value of `max_dec_frame_buffering` shall be inferred to be equal to `MaxDpbSize`.

E.2.2 HRD parameters semantics

`cpb_cnt_minus1` plus 1 specifies the number of alternative CPB specifications in the bitstream. The value of `cpb_cnt_minus1` shall be in the range of 0 to 31, inclusive. When `low_delay_hrd_flag` is equal to 1, `cpb_cnt_minus1` shall be equal to 0. When `cpb_cnt_minus1` is not present, it shall be inferred to be equal to 0.

`bit_rate_scale` (together with `bit_rate_value_minus1[SchedSelIdx]`) specifies the maximum input bit rate of the `SchedSelIdx`-th CPB.

`cpb_size_scale` (together with `cpb_size_value_minus1[SchedSelIdx]`) specifies the CPB size of the `SchedSelIdx`-th CPB.

`bit_rate_value_minus1[SchedSelIdx]` (together with `bit_rate_scale`) specifies the maximum input bit rate for the `SchedSelIdx`-th CPB. `bit_rate_value_minus1[SchedSelIdx]` shall be in the range of 0 to $2^{32} - 2$, inclusive. For any `SchedSelIdx > 0`, `bit_rate_value_minus1[SchedSelIdx]` shall be greater than `bit_rate_value_minus1[SchedSelIdx - 1]`. The bit rate in bits per second is given by

$$\text{BitRate[SchedSelIdx]} = (\text{bit_rate_value_minus1[SchedSelIdx]} + 1) * 2^{(6 + \text{bit_rate_scale})} \quad (\text{E-37})$$

When the `bit_rate_value_minus1[SchedSelIdx]` syntax element is not present, the value of `BitRate[SchedSelIdx]` shall be inferred as follows.

- If `profile_idc` is equal to 66, 77, or 88, `BitRate[SchedSelIdx]` shall be inferred to be equal to $1000 * \text{MaxBR}$ for VCL HRD parameters and to be equal to $1200 * \text{MaxBR}$ for NAL HRD parameters, where `MaxBR` is specified in subclause A.3.1.
- Otherwise, `BitRate[SchedSelIdx]` shall be inferred to be equal to `cpbBrVclFactor * MaxBR` for VCL HRD parameters and to be equal to `cpbBrNalFactor * MaxBR` for NAL HRD parameters, where `cpbBrVclFactor`, `cpbBrNalFactor`, and `MaxBR` are specified in subclause A.3.3.

`cpb_size_value_minus1[SchedSelIdx]` is used together with `cpb_size_scale` to specify the `SchedSelIdx`-th CPB size. `cpb_size_value_minus1[SchedSelIdx]` shall be in the range of 0 to $2^{32} - 2$, inclusive. For any `SchedSelIdx` greater than 0, `cpb_size_value_minus1[SchedSelIdx]` shall be less than or equal to `cpb_size_value_minus1[SchedSelIdx - 1]`.

The CPB size in bits is given by

$$\text{CpbSize[SchedSelIdx]} = (\text{cpb_size_value_minus1[SchedSelIdx]} + 1) * 2^{(4 + \text{cpb_size_scale})} \quad (\text{E-38})$$

When the `cpb_size_value_minus1[SchedSelIdx]` syntax element is not present, the value of `CpbSize[SchedSelIdx]` shall be inferred as follows.

- If `profile_idc` is equal to 66, 77, or 88, `CpbSize[SchedSelIdx]` shall be inferred to be equal to $1000 * \text{MaxCPB}$ for VCL HRD parameters and to be equal to $1200 * \text{MaxCPB}$ for NAL HRD parameters, where `MaxCPB` is specified in subclause A.3.1.
- Otherwise, `CpbSize[SchedSelIdx]` shall be inferred to be equal to `cpbBrVclFactor * MaxCPB` for VCL HRD parameters and to be equal to `cpbBrNalFactor * MaxCPB` for NAL HRD parameters, where `cpbBrVclFactor`, `cpbBrNalFactor`, and `MaxCPB` are specified in subclause A.3.3.

`cbr_flag[SchedSelIdx]` equal to 0 specifies that to decode this bitstream by the HRD using the `SchedSelIdx`-th CPB specification, the hypothetical stream delivery scheduler (HSS) operates in an intermittent bit rate mode. `cbr_flag[SchedSelIdx]` equal to 1 specifies that the HSS operates in a constant bit rate (CBR) mode. When the `cbr_flag[SchedSelIdx]` syntax element is not present, the value of `cbr_flag` shall be inferred to be equal to 0.

`initial_cpb_removal_delay_length_minus1` specifies the length in bits of the `initial_cpb_removal_delay[SchedSelIdx]` and `initial_cpb_removal_delay_offset[SchedSelIdx]` syntax elements of the buffering period SEI message. The length of `initial_cpb_removal_delay[SchedSelIdx]` and of `initial_cpb_removal_delay_offset[SchedSelIdx]` is `initial_cpb_removal_delay_length_minus1 + 1`. When the `initial_cpb_removal_delay_length_minus1` syntax element is present in more than one `hrd_parameters()` syntax structure within the VUI parameters syntax structure, the value of the `initial_cpb_removal_delay_length_minus1` parameters shall be equal in both `hrd_parameters()` syntax structures. When the `initial_cpb_removal_delay_length_minus1` syntax element is not present, it shall be inferred to be equal to 23.

`cpb_removal_delay_length_minus1` specifies the length in bits of the `cpb_removal_delay` syntax element. The length of the `cpb_removal_delay` syntax element of the picture timing SEI message is `cpb_removal_delay_length_minus1 + 1`. When the `cpb_removal_delay_length_minus1` syntax element is present in more than one `hrd_parameters()` syntax structure within the VUI parameters syntax structure, the value of the `cpb_removal_delay_length_minus1` parameters shall be equal in both `hrd_parameters()` syntax structures. When the `cpb_removal_delay_length_minus1` syntax element is not present, it shall be inferred to be equal to 23.

`dpb_output_delay_length_minus1` specifies the length in bits of the `dpb_output_delay` syntax element. The length of the `dpb_output_delay` syntax element of the picture timing SEI message is `dpb_output_delay_length_minus1 + 1`. When the `dpb_output_delay_length_minus1` syntax element is present in more than one `hrd_parameters()` syntax structure within the VUI parameters syntax structure, the value of the `dpb_output_delay_length_minus1` parameters shall be equal in both `hrd_parameters()` syntax structures. When the `dpb_output_delay_length_minus1` syntax element is not present, it shall be inferred to be equal to 23.

`time_offset_length` greater than 0 specifies the length in bits of the `time_offset` syntax element. `time_offset_length` equal to 0 specifies that the `time_offset` syntax element is not present. When the `time_offset_length` syntax element is present in more than one `hrd_parameters()` syntax structure within the VUI parameters syntax structure, the value of the `time_offset_length` parameters shall be equal in both `hrd_parameters()` syntax structures. When the `time_offset_length` syntax element is not present, it shall be inferred to be equal to 24.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	General tariff principles
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects and next-generation networks
Series Z	Languages and general software aspects for telecommunication systems

EXHIBIT 3

I n t e r n a t i o n a l T e l e c o m m u n i c a t i o n U n i o n

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

H.265

(08/2021)

SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS

Infrastructure of audiovisual services – Coding of moving
video

High efficiency video coding

Recommendation ITU-T H.265

ITU-T H-SERIES RECOMMENDATIONS
AUDIOVISUAL AND MULTIMEDIA SYSTEMS

CHARACTERISTICS OF VISUAL TELEPHONE SYSTEMS	H.100–H.199
INFRASTRUCTURE OF AUDIOVISUAL SERVICES	
General	H.200–H.219
Transmission multiplexing and synchronization	H.220–H.229
Systems aspects	H.230–H.239
Communication procedures	H.240–H.259
Coding of moving video	H.260–H.279
Related systems aspects	H.280–H.299
Systems and terminal equipment for audiovisual services	H.300–H.349
Directory services architecture for audiovisual and multimedia services	H.350–H.359
Quality of service architecture for audiovisual and multimedia services	H.360–H.369
Telepresence, immersive environments, virtual and extended reality	H.420–H.439
Supplementary services for multimedia	H.450–H.499
MOBILITY AND COLLABORATION PROCEDURES	
Overview of Mobility and Collaboration, definitions, protocols and procedures	H.500–H.509
Mobility for H-Series multimedia systems and services	H.510–H.519
Mobile multimedia collaboration applications and services	H.520–H.529
Security for mobile multimedia systems and services	H.530–H.539
Security for mobile multimedia collaboration applications and services	H.540–H.549
VEHICULAR GATEWAYS AND INTELLIGENT TRANSPORTATION SYSTEMS (ITS)	
Architecture for vehicular gateways	H.550–H.559
Vehicular gateway interfaces	H.560–H.569
BROADBAND, TRIPLE-PLAY AND ADVANCED MULTIMEDIA SERVICES	
Broadband multimedia services over VDSL	H.610–H.619
Advanced multimedia services and applications	H.620–H.629
Content delivery and ubiquitous sensor network applications	H.640–H.649
IPTV MULTIMEDIA SERVICES AND APPLICATIONS FOR IPTV	
General aspects	H.700–H.719
IPTV terminal devices	H.720–H.729
IPTV middleware	H.730–H.739
IPTV application event handling	H.740–H.749
IPTV metadata	H.750–H.759
IPTV multimedia application frameworks	H.760–H.769
IPTV service discovery up to consumption	H.770–H.779
Digital Signage	H.780–H.789
E-HEALTH MULTIMEDIA SYSTEMS, SERVICES AND APPLICATIONS	
Personal health systems	H.810–H.819
Interoperability compliance testing of personal health systems (HRN, PAN, LAN, TAN and WAN)	H.820–H.859
Multimedia e-health data exchange services	H.860–H.869
Safe listening	H.870–H.879

For further details, please refer to the list of ITU-T Recommendations.

High efficiency video coding

Summary

Recommendation ITU-T H.265 | International Standard ISO/IEC 23008-2 represents an evolution of the existing video coding Recommendations (ITU-T H.261, ITU-T H.262, ITU-T H.263 and ITU-T H.264) and was developed in response to the growing need for higher compression of moving pictures for various applications such as Internet streaming, communication, videoconferencing, digital storage media and television broadcasting. It is also designed to enable the use of the coded video representation in a flexible manner for a wide variety of network environments. The use of this Recommendation | International Standard allows motion video to be manipulated as a form of computer data and to be stored on various storage media, transmitted and received over existing and future networks and distributed on existing and future broadcasting channels.

This revision adds an additional SEI message for shutter interval information, and also includes corrections to various minor defects in the prior content of the Specification.

This Recommendation | International Standard was developed jointly with ISO/IEC JTC 1/SC 29 and corresponds in a technically aligned manner to ISO/IEC 23008-2.

History

Edition	Recommendation	Approval	Study Group	Unique ID*
1.0	ITU-T H.265	2013-04-13	16	11.1002/1000/11885
2.0	ITU-T H.265 (V2)	2014-10-29	16	11.1002/1000/12296
3.0	ITU-T H.265 (V3)	2015-04-29	16	11.1002/1000/12455
4.0	ITU-T H.265 (V4)	2016-12-22	16	11.1002/1000/12905
5.0	ITU-T H.265 (V5)	2018-02-13	16	11.1002/1000/13433
6.0	ITU-T H.265 (V6)	2019-06-29	16	11.1002/1000/13904
7.0	ITU-T H.265 (V7)	2019-11-29	16	11.1002/1000/14107
8.0	ITU-T H.265 (V8)	2021-08-22	16	11.1002/1000/14660

Keywords

Compression coding, digital video, image coding, supplemental enhancement information, video coding, video compression, visual coding.

* To access the Recommendation, type the URL <http://handle.itu.int/> in the address field of your web browser, followed by the Recommendation's unique ID. For example, <http://handle.itu.int/11.1002/1000/11830-en>.

FOREWORD

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a worldwide basis.

The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups which, in turn, produce Recommendations on these topics.

The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1.

In some areas of information technology which fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

NOTE

In this Recommendation, the expression "Administration" is used for conciseness to indicate both a telecommunication administration and a recognized operating agency.

Compliance with this Recommendation is voluntary. However, the Recommendation may contain certain mandatory provisions (to ensure, e.g., interoperability or applicability) and compliance with the Recommendation is achieved when all of these mandatory provisions are met. The words "shall" or some other obligatory language such as "must" and the negative equivalents are used to express requirements. The use of such words does not suggest that compliance with the Recommendation is required of any party.

INTELLECTUAL PROPERTY RIGHTS

ITU draws attention to the possibility that the practice or implementation of this Recommendation may involve the use of a claimed Intellectual Property Right. ITU takes no position concerning the evidence, validity or applicability of claimed Intellectual Property Rights, whether asserted by ITU members or others outside of the Recommendation development process.

As of the date of approval of this Recommendation, ITU had received notice of intellectual property, protected by patents/software copyrights, which may be required to implement this Recommendation. However, implementers are cautioned that this may not represent the latest information and are therefore strongly urged to consult the appropriate ITU-T databases available via the ITU-T website at <http://www.itu.int/ITU-T/ipr/>.

© ITU 2021

All rights reserved. No part of this publication may be reproduced, by any means whatsoever, without the prior written permission of ITU.

TABLE OF CONTENTS

	<i>Page</i>
0	Introduction 1
0.1	General 1
0.2	Prologue 1
0.3	Purpose..... 1
0.4	Applications 1
0.5	Publication and versions of this Specification 2
0.6	Profiles, tiers and levels 2
0.7	Overview of the design characteristics..... 3
0.8	How to read this Specification 3
1	Scope..... 3
2	Normative references 3
2.1	General..... 3
2.2	Identical Recommendations International Standards 4
2.3	Paired Recommendations International Standards equivalent in technical content 4
2.4	Additional references 4
3	Definitions..... 4
4	Abbreviations and acronyms 13
5	Conventions..... 16
5.1	General..... 16
5.2	Arithmetic operators 16
5.3	Logical operators 16
5.4	Relational operators 16
5.5	Bit-wise operators 16
5.6	Assignment operators..... 17
5.7	Range notation 17
5.8	Mathematical functions 17
5.9	Order of operation precedence 18
5.10	Variables, syntax elements and tables..... 19
5.11	Text description of logical operations..... 20
5.12	Processes 21
6	Bitstream and picture formats, partitionings, scanning processes and neighbouring relationships 21
6.1	Bitstream formats 21
6.2	Source, decoded and output picture formats 21
6.3	Partitioning of pictures, slices, slice segments, tiles, CTUs and CTBs 23
6.3.1	Partitioning of pictures into slices, slice segments and tiles 23
6.3.2	Block and quadtree structures 24
6.3.3	Spatial or component-wise partitionings..... 25
6.4	Availability processes 26
6.4.1	Derivation process for z-scan order block availability..... 26
6.4.2	Derivation process for prediction block availability 26
6.5	Scanning processes 27
6.5.1	CTB raster and tile scanning conversion process..... 27
6.5.2	Z-scan order array initialization process 29
6.5.3	Up-right diagonal scan order array initialization process..... 29
6.5.4	Horizontal scan order array initialization process..... 29
6.5.5	Vertical scan order array initialization process 30
6.5.6	Traverse scan order array initialization process 30
7	Syntax and semantics 30
7.1	Method of specifying syntax in tabular form 30
7.2	Specification of syntax functions and descriptors 31
7.3	Syntax in tabular form..... 33
7.3.1	NAL unit syntax..... 33
7.3.2	Raw byte sequence payloads, trailing bits and byte alignment syntax..... 34

7.3.3	Profile, tier and level syntax	42
7.3.4	Scaling list data syntax.....	45
7.3.5	Supplemental enhancement information message syntax	45
7.3.6	Slice segment header syntax	46
7.3.7	Short-term reference picture set syntax.....	50
7.3.8	Slice segment data syntax	51
7.4	Semantics	64
7.4.1	General.....	64
7.4.2	NAL unit semantics	64
7.4.3	Raw byte sequence payloads, trailing bits and byte alignment semantics	72
7.4.4	Profile, tier and level semantics	89
7.4.5	Scaling list data semantics	92
7.4.6	Supplemental enhancement information message semantics.....	95
7.4.7	Slice segment header semantics	95
7.4.8	Short-term reference picture set semantics	102
7.4.9	Slice segment data semantics.....	104
8	Decoding process	117
8.1	General decoding process	117
8.1.1	General.....	117
8.1.2	CVSG decoding process	117
8.1.3	Decoding process for a coded picture with nuh_layer_id equal to 0.....	117
8.2	NAL unit decoding process.....	119
8.3	Slice decoding process	119
8.3.1	Decoding process for picture order count	119
8.3.2	Decoding process for reference picture set	120
8.3.3	Decoding process for generating unavailable reference pictures	124
8.3.4	Decoding process for reference picture lists construction.....	125
8.3.5	Decoding process for collocated picture and no backward prediction flag.....	126
8.4	Decoding process for coding units coded in intra prediction mode	126
8.4.1	General decoding process for coding units coded in intra prediction mode.....	126
8.4.2	Derivation process for luma intra prediction mode.....	130
8.4.3	Derivation process for chroma intra prediction mode.....	132
8.4.4	Decoding process for intra blocks.....	133
8.5	Decoding process for coding units coded in inter prediction mode	143
8.5.1	General decoding process for coding units coded in inter prediction mode.....	143
8.5.2	Inter prediction process.....	143
8.5.3	Decoding process for prediction units in inter prediction mode	146
8.5.4	Decoding process for the residual signal of coding units coded in inter prediction mode	171
8.6	Scaling, transformation and array construction process prior to deblocking filter process.....	175
8.6.1	Derivation process for quantization parameters.....	175
8.6.2	Scaling and transformation process	176
8.6.3	Scaling process for transform coefficients	178
8.6.4	Transformation process for scaled transform coefficients	179
8.6.5	Residual modification process for blocks using a transform bypass.....	181
8.6.6	Residual modification process for transform blocks using cross-component prediction	181
8.6.7	Picture construction process prior to in-loop filter process.....	182
8.6.8	Residual modification process for blocks using adaptive colour transform.....	182
8.7	In-loop filter process	184
8.7.1	General.....	184
8.7.2	Deblocking filter process	185
8.7.3	Sample adaptive offset process	198
9	Parsing process.....	200
9.1	General.....	200
9.2	Parsing process for 0-th order Exp-Golomb codes	201
9.2.1	General.....	201
9.2.2	Mapping process for signed Exp-Golomb codes	202
9.3	CABAC parsing process for slice segment data	203
9.3.1	General.....	203
9.3.2	Initialization process	205
9.3.3	Binarization process.....	218

9.3.4	Decoding process flow.....	227
9.3.5	Arithmetic encoding process (informative).....	241
10	Sub-bitstream extraction process.....	247
Annex A	Profiles, tiers and levels	248
A.1	Overview of profiles, tiers and levels.....	248
A.2	Requirements on video decoder capability	248
A.3	Profiles	248
A.3.1	General.....	248
A.3.2	Main profile	248
A.3.3	Main 10 and Main 10 Still Picture profiles.....	249
A.3.4	Main Still Picture profile.....	250
A.3.5	Format range extensions profiles	251
A.3.6	High throughput profiles.....	255
A.3.7	Screen content coding extensions profiles	257
A.3.8	High throughput screen content coding extensions profiles.....	260
A.4	Tiers and levels	263
A.4.1	General tier and level limits	263
A.4.2	Profile-specific level limits for the video profiles.....	264
A.4.3	Effect of level limits on picture rate for the video profiles (informative)	268
Annex B	Byte stream format	272
B.1	General.....	272
B.2	Byte stream NAL unit syntax and semantics	272
B.2.1	Byte stream NAL unit syntax.....	272
B.2.2	Byte stream NAL unit semantics	272
B.3	Byte stream NAL unit decoding process.....	273
B.4	Decoder byte-alignment recovery (informative).....	273
Annex C	Hypothetical reference decoder	274
C.1	General.....	274
C.2	Operation of coded picture buffer	278
C.2.1	General.....	278
C.2.2	Timing of decoding unit arrival	278
C.2.3	Timing of decoding unit removal and decoding of decoding unit	280
C.3	Operation of the decoded picture buffer	283
C.3.1	General.....	283
C.3.2	Removal of pictures from the DPB before decoding of the current picture	283
C.3.3	Picture output.....	284
C.3.4	Current decoded picture marking and storage.....	284
C.3.5	Removal of pictures from the DPB after decoding of the current picture.....	285
C.4	Bitstream conformance	285
C.5	Decoder conformance	286
C.5.1	General.....	286
C.5.2	Operation of the output order DPB	287
Annex D	Supplemental enhancement information	290
D.1	General.....	290
D.2	SEI payload syntax.....	290
D.2.1	General SEI message syntax	290
D.2.2	Buffering period SEI message syntax	294
D.2.3	Picture timing SEI message syntax	295
D.2.4	Pan-scan rectangle SEI message syntax.....	295
D.2.5	Filler payload SEI message syntax	296
D.2.6	User data registered by Recommendation ITU-T T.35 SEI message syntax	296
D.2.7	User data unregistered SEI message syntax	296
D.2.8	Recovery point SEI message syntax	296
D.2.9	Scene information SEI message syntax	297
D.2.10	Picture snapshot SEI message syntax	297
D.2.11	Progressive refinement segment start SEI message syntax.....	297
D.2.12	Progressive refinement segment end SEI message syntax.....	297
D.2.13	Film grain characteristics SEI message syntax	298
D.2.14	Post-filter hint SEI message syntax.....	298

D.2.15	Tone mapping information SEI message syntax	299
D.2.16	Frame packing arrangement SEI message syntax	300
D.2.17	Display orientation SEI message syntax	300
D.2.18	Green metadata SEI message syntax	300
D.2.19	Structure of pictures information SEI message syntax	301
D.2.20	Decoded picture hash SEI message syntax	301
D.2.21	Active parameter sets SEI message syntax	301
D.2.22	Decoding unit information SEI message syntax	302
D.2.23	Temporal sub-layer zero index SEI message syntax	302
D.2.24	Scalable nesting SEI message syntax	302
D.2.25	Region refresh information SEI message syntax	303
D.2.26	No display SEI message syntax	303
D.2.27	Time code SEI message syntax	303
D.2.28	Mastering display colour volume SEI message syntax	304
D.2.29	Segmented rectangular frame packing arrangement SEI message syntax	304
D.2.30	Temporal motion-constrained tile sets SEI message syntax	305
D.2.31	Chroma resampling filter hint SEI message syntax	306
D.2.32	Knee function information SEI message syntax	306
D.2.33	Colour remapping information SEI message syntax	307
D.2.34	Deinterlaced field identification SEI message syntax	308
D.2.35	Content light level information SEI message syntax	308
D.2.36	Dependent random access point indication SEI message syntax	308
D.2.37	Coded region completion SEI message syntax	308
D.2.38	Alternative transfer characteristics information SEI message syntax	308
D.2.39	Ambient viewing environment SEI message syntax	308
D.2.40	Content colour volume SEI message syntax	309
D.2.41	Syntax of omnidirectional video specific SEI messages	309
D.2.42	Regional nesting SEI message syntax	313
D.2.43	Motion-constrained tile sets extraction information sets SEI message syntax	314
D.2.44	Motion-constrained tile sets extraction information nesting SEI message syntax	315
D.2.45	SEI manifest SEI message syntax	315
D.2.46	SEI prefix indication SEI message syntax	315
D.2.47	Annotated regions SEI message syntax	316
D.2.48	Shutter interval information SEI message syntax	317
D.2.49	Reserved SEI message syntax	317
D.3	SEI payload semantics	317
D.3.1	General SEI payload semantics	317
D.3.2	Buffering period SEI message semantics	322
D.3.3	Picture timing SEI message semantics	324
D.3.4	Pan-scan rectangle SEI message semantics	329
D.3.5	Filler payload SEI message semantics	330
D.3.6	User data registered by Recommendation ITU-T T.35 SEI message semantics	330
D.3.7	User data unregistered SEI message semantics	331
D.3.8	Recovery point SEI message semantics	331
D.3.9	Scene information SEI message semantics	332
D.3.10	Picture snapshot SEI message semantics	334
D.3.11	Progressive refinement segment start SEI message semantics	334
D.3.12	Progressive refinement segment end SEI message semantics	335
D.3.13	Film grain characteristics SEI message semantics	335
D.3.14	Post-filter hint SEI message semantics	341
D.3.15	Tone mapping information SEI message semantics	342
D.3.16	Frame packing arrangement SEI message semantics	346
D.3.17	Display orientation SEI message semantics	353
D.3.18	Green metadata SEI message semantics	354
D.3.19	Structure of pictures information SEI message semantics	354
D.3.20	Decoded picture hash SEI message semantics	355
D.3.21	Active parameter sets SEI message semantics	356
D.3.22	Decoding unit information SEI message semantics	357
D.3.23	Temporal sub-layer zero index SEI message semantics	358
D.3.24	Scalable nesting SEI message semantics	359
D.3.25	Region refresh information SEI message semantics	360
D.3.26	No display SEI message semantics	361
D.3.27	Time code SEI message semantics	362

D.3.28	Mastering display colour volume SEI message semantics.....	364
D.3.29	Segmented rectangular frame packing arrangement SEI message semantics	365
D.3.30	Temporal motion-constrained tile sets SEI message semantics	368
D.3.31	Chroma resampling filter hint SEI message semantics	372
D.3.32	Knee function information SEI message semantics	381
D.3.33	Colour remapping information SEI message semantics.....	382
D.3.34	Deinterlaced field identification SEI message semantics.....	385
D.3.35	Content light level information SEI message semantics	385
D.3.36	Dependent random access point indication SEI message semantics.....	385
D.3.37	Coded region completion SEI message semantics	386
D.3.38	Alternative transfer characteristics SEI message semantics.....	386
D.3.39	Ambient viewing environment SEI message semantics.....	386
D.3.40	Content colour volume SEI message semantics.....	387
D.3.41	Semantics of omnidirectional video specific SEI messages	389
D.3.42	Regional nesting SEI message semantics	406
D.3.43	Motion-constrained tile sets extraction information sets SEI message semantics	408
D.3.44	Motion-constrained tile sets extraction information nesting SEI message semantics	410
D.3.45	SEI manifest SEI message semantics.....	411
D.3.46	SEI prefix indication SEI message semantics	412
D.3.47	Annotated regions SEI message semantics	413
D.3.48	Shutter interval information SEI message semantics	415
D.3.49	Reserved SEI message semantics.....	416
Annex E	Video usability information	417
E.1	General.....	417
E.2	VUI syntax	417
E.2.1	VUI parameters syntax	417
E.2.2	HRD parameters syntax	419
E.2.3	Sub-layer HRD parameters syntax.....	420
E.3	VUI semantics.....	420
E.3.1	VUI parameters semantics	420
E.3.2	HRD parameters semantics	437
E.3.3	Sub-layer HRD parameters semantics	440
Annex F	Common specifications for multi-layer extensions.....	441
F.1	Scope.....	441
F.2	Normative references	441
F.3	Definitions.....	441
F.4	Abbreviations.....	443
F.5	Conventions	444
F.6	Bitstream and picture formats, partitionings, scanning processes and neighbouring relationships.....	444
F.7	Syntax and semantics	444
F.7.1	Method of specifying syntax in tabular form.....	444
F.7.2	Specification of syntax functions, categories and descriptors.....	444
F.7.3	Syntax in tabular form	444
F.7.4	Semantics	461
F.8	Decoding process	498
F.8.1	General decoding process	498
F.8.2	NAL unit decoding process.....	506
F.8.3	Slice decoding processes.....	506
F.8.4	Decoding process for coding units coded in intra prediction mode	510
F.8.5	Decoding process for coding units coded in inter prediction mode	510
F.8.6	Scaling, transformation and array construction process prior to deblocking filter process	510
F.8.7	In-loop filter process	510
F.9	Parsing process.....	510
F.10	Specification of bitstream subsets.....	511
F.10.1	Sub-bitstream extraction process	511
F.10.2	Independent non-base layer rewriting process	511
F.10.3	Sub-bitstream extraction process for additional layer sets	512
F.11	Profiles, tiers and levels	512
F.11.1	Independent non-base layer decoding capability	512

F.11.2	Decoder capabilities.....	513
F.11.3	Derivation of sub-bitstreams subBitstream and baseBitstream.....	514
F.12	Byte stream format.....	514
F.13	Hypothetical reference decoder.....	514
F.13.1	General.....	514
F.13.2	Operation of bitstream partition buffer	519
F.13.3	Operation of decoded picture buffer	525
F.13.4	Bitstream conformance	527
F.13.5	Decoder conformance	529
F.13.6	Demultiplexing process for deriving a bitstream partition.....	532
F.14	Supplemental enhancement information	533
F.14.1	General.....	533
F.14.2	SEI payload syntax	533
F.14.3	SEI payload semantics	537
F.15	Video usability information	555
F.15.1	General.....	555
F.15.2	VUI syntax.....	555
F.15.3	VUI semantics.....	555
Annex G	Multiview high efficiency video coding	557
G.1	Scope.....	557
G.2	Normative references	557
G.3	Definitions.....	557
G.4	Abbreviations.....	557
G.5	Conventions	557
G.6	Bitstream and picture formats, partitionings, scanning processes, and neighbouring relationships	557
G.7	Syntax and semantics.....	557
G.8	Decoding processes.....	557
G.8.1	General decoding process	557
G.8.2	NAL unit decoding process.....	558
G.8.3	Slice decoding processes.....	558
G.8.4	Decoding process for coding units coded in intra prediction mode	558
G.8.5	Decoding process for coding units coded in inter prediction mode	558
G.8.6	Scaling, transformation and array construction process prior to deblocking filter process	558
G.8.7	In-loop filter process	558
G.9	Parsing process.....	559
G.10	Specification of bitstream subsets.....	559
G.11	Profiles, tiers and levels	559
G.11.1	Profiles	559
G.11.2	Tiers and levels	560
G.11.3	Decoder capabilities.....	563
G.12	Byte stream format.....	563
G.13	Hypothetical reference decoder.....	563
G.14	Supplemental enhancement information	563
G.14.1	General.....	563
G.14.2	SEI payload syntax	563
G.14.3	SEI payload semantics	567
G.15	Video usability information	576
Annex H	Scalable high efficiency video coding.....	577
H.1	Scope.....	577
H.2	Normative references	577
H.3	Definitions.....	577
H.4	Abbreviations.....	577
H.5	Conventions	577
H.6	Bitstream and picture formats, partitionings, scanning processes, and neighbouring relationships	577
H.7	Syntax and semantics.....	577
H.8	Decoding processes.....	577

H.8.1	General decoding process	577
H.8.2	NAL unit decoding process.....	592
H.8.3	Slice decoding processes.....	592
H.8.4	Decoding process for coding units coded in intra prediction mode	592
H.8.5	Decoding process for coding units coded in inter prediction mode	592
H.8.6	Scaling, transformation and array construction process prior to deblocking filter process	593
H.8.7	In-loop filter process	593
H.9	Parsing process.....	593
H.10	Specification of bitstream subsets.....	593
H.11	Profiles, tiers and levels	593
H.11.1	Profiles	593
H.11.2	Tiers and levels	597
H.11.3	Decoder capabilities	600
H.12	Byte stream format.....	600
H.13	Hypothetical reference decoder.....	600
H.14	Supplemental enhancement information	600
H.15	Video usability information	600
Annex I	3D high efficiency video coding.....	601
I.1	Scope.....	601
I.2	Normative references	601
I.3	Definitions.....	601
I.4	Abbreviations.....	601
I.5	Conventions	601
I.6	Bitstream and picture formats, partitionings, scanning processes, and neighbouring relationships	601
I.6.1	Bitstream formats.....	601
I.6.2	Source, decoded, and output picture formats	602
I.6.3	Partitioning of pictures, slices, slice segments, tiles, CTUs, and CTBs.....	602
I.6.4	Availability processes	602
I.6.5	Scanning processes	602
I.6.6	Derivation process for a wedgelet partition pattern table.....	602
I.7	Syntax and semantics	604
I.7.1	Method of specifying syntax in tabular form	604
I.7.2	Specification of syntax functions, categories, and descriptors.....	604
I.7.3	Syntax in tabular form	604
I.7.4	Semantics	617
I.8	Decoding process	632
I.8.1	General decoding process	632
I.8.2	NAL unit decoding process.....	632
I.8.3	Slice decoding process	632
I.8.4	Decoding process for coding units coded in intra prediction mode	634
I.8.5	Decoding process for coding units coded in inter prediction mode	642
I.8.6	Scaling, transformation and array construction process prior to deblocking filter process	675
I.8.7	In-loop filter process	675
I.9	Parsing process.....	675
I.9.1	General.....	675
I.9.2	Parsing process for 0-th order Exp-Golomb codes	676
I.9.3	CABAC parsing process for slice segment data	676
I.10	Specification of bitstream subsets.....	682
I.11	Profiles, tiers, and levels	682
I.11.1	Profiles	682
I.11.2	Tiers and levels	684
I.11.3	Decoder capabilities.....	684
I.12	Byte stream format.....	684
I.13	Hypothetical reference decoder.....	684
I.14	Supplemental enhancement information	684
I.14.1	General.....	684
I.14.2	SEI payload syntax	684

I.14.3	SEI payload semantics	686
I.15	Video usability information	692
Bibliography		693

LIST OF FIGURES

Figure 6-1 – Nominal vertical and horizontal locations of 4:2:0 luma and chroma samples in a picture	22
Figure 6-2 – Nominal vertical and horizontal locations of 4:2:2 luma and chroma samples in a picture	23
Figure 6-3 – Nominal vertical and horizontal locations of 4:4:4 luma and chroma samples in a picture	23
Figure 6-4 – A picture with 11 by 9 luma CTBs that is partitioned into two slices, the first of which is partitioned into three slice segments (informative)	24
Figure 6-5 – A picture with 11 by 9 luma CTBs that is partitioned into two tiles and one slice (left) or is partitioned into two tiles and three slices (right) (informative).....	24
Figure 7-1 – Structure of an access unit not containing any NAL units with nal_unit_type equal to FD_NUT, SUFFIX_SEI_NUT, VPS_NUT, SPS_NUT, PPS_NUT, RSV_VCL_N10, RSV_VCL_R11, RSV_VCL_N12, RSV_VCL_R13, RSV_VCL_N14, RSV_VCL_R15, RSV_IRAP_VCL22 or RSV_IRAP_VCL23, or in the range of RSV_VCL24..RSV_VCL31, RSV_NVCL41..RSV_NVCL47 or UNSPEC48..UNSPEC63	72
Figure 8-1 – Intra prediction mode directions (informative)	130
Figure 8-2 – Intra prediction angle definition (informative).....	139
Figure 8-3 – Spatial motion vector neighbours (informative)	158
Figure 8-4 – Integer samples (shaded blocks with upper-case letters) and fractional sample positions (un-shaded blocks with lower-case letters) for quarter sample luma interpolation.....	166
Figure 8-5 – Integer samples (shaded blocks with upper-case letters) and fractional sample positions (un-shaded blocks with lower-case letters) for eighth sample chroma interpolation	168
Figure 9-1 – Illustration of CABAC parsing process for a syntax element synEl (informative)	204
Figure 9-2 – Spatial neighbour T that is used to invoke the CTB availability derivation process relative to the current CTB (informative).....	205
Figure 9-3 – Illustration of CABAC initialization process (informative).....	206
Figure 9-4 – Illustration of CABAC storage process (informative).....	217
Figure 9-5 – Overview of the arithmetic decoding process for a single bin (informative)	235
Figure 9-6 – Flowchart for decoding a decision	237
Figure 9-7 – Flowchart of renormalization	239
Figure 9-8 – Flowchart of bypass decoding process	240
Figure 9-9 – Flowchart of decoding a decision before termination	241
Figure 9-10 – Flowchart for encoding a decision	243
Figure 9-11 – Flowchart of renormalization in the encoder	244
Figure 9-12 – Flowchart of PutBit(B).....	244
Figure 9-13 – Flowchart of encoding bypass.....	245
Figure 9-14 – Flowchart of encoding a decision before termination	246
Figure 9-15 – Flowchart of flushing at termination.....	246
Figure C.1 – Structure of byte streams and NAL unit streams for HRD conformance checks.....	274
Figure C.2 – HRD buffer model	277
Figure D.1 – Nominal vertical and horizontal sampling locations of 4:2:0 samples in top and bottom fields	327
Figure D.2 – Nominal vertical and horizontal sampling locations of 4:2:2 samples in top and bottom fields	327
Figure D.3 – Nominal vertical and horizontal sampling locations of 4:4:4 samples in top and bottom fields	327
Figure D.4 – Rearrangement and upconversion of side-by-side packing arrangement with frame_packing_arrangement_type equal to 3, quincunx_sampling_flag equal to 0 and (x, y) equal to (0, 0) or (4, 8) for both constituent frames.....	350

Figure D.5 – Rearrangement and upconversion of side-by-side packing arrangement with frame_packing_arrangement_type equal to 3, quincunx_sampling_flag equal to 0, (x, y) equal to (12, 8) for constituent frame 0 and (x, y) equal to (0, 0) or (4, 8) for constituent frame 1	351
Figure D.6 – Rearrangement and upconversion of top-bottom packing arrangement with frame_packing_arrangement_type equal to 4, quincunx_sampling_flag equal to 0 and (x, y) equal to (0, 0) or (8, 4) for both constituent frames.....	351
Figure D.7 – Rearrangement and upconversion of top-bottom packing arrangement with frame_packing_arrangement_type equal to 4, quincunx_sampling_flag equal to 0, (x, y) equal to (8, 12) for constituent frame 0 and (x, y) equal to (0, 0) or (8, 4) for constituent frame 1	352
Figure D.8 – Rearrangement and upconversion of side-by-side packing arrangement with quincunx sampling (frame_packing_arrangement_type equal to 3 with quincunx_sampling_flag equal to 1)	352
Figure D.9 – Rearrangement of a temporal interleaving frame arrangement (frame_packing_arrangement_type equal to 5).....	353
Figure D.10 – Rearrangement of a segmented rectangular frame packing arrangement	368
Figure D.11 – A knee function with num_knee_points_minus1 equal to 2.....	381
Figure E.1 – Location of chroma samples for top and bottom fields for chroma_format_idc equal to 1 (4:2:0 chroma format) as a function of chroma_sample_loc_type_top_field and chroma_sample_loc_type_bottom_field.....	432
Figure E.2 – Location of the top-left chroma sample when chroma_format_idc is equal to 1 (4:2:0 chroma format) as a function of ChromaLocType	433
Figure E.3 – Location of the top-left chroma sample when chroma_format_idc is equal to 1 (4:2:0 chroma format) when ChromaLocType is equal to 1	433
Figure F.1 – Bitstream-partition-specific HRD buffer model.....	518

LIST OF TABLES

Table 5-1 – Operation precedence from highest (at top of table) to lowest (at bottom of table)	19
Table 6-1 – SubWidthC and SubHeightC values derived from chroma_format_idc and separate_colour_plane_flag	22
Table 7-1 – NAL unit type codes and NAL unit type classes.....	66
Table 7-2 – Interpretation of pic_type	88
Table 7-3 – Specification of sizeId	93
Table 7-4 – Specification of matrixId according to sizeId, prediction mode and colour component	93
Table 7-5 – Specification of default values of ScalingList[0][matrixId][i] with i = 0..15	94
Table 7-6 – Specification of default values of ScalingList[1..3][matrixId][i] with i = 0..63	94
Table 7-7 – Name association to slice_type	96
Table 7-8 – Specification of the SAO type.....	105
Table 7-9 – Specification of the SAO edge offset class	106
Table 7-10 – Name association to prediction mode and partitioning type.....	108
Table 7-11 – Name association to inter prediction mode	109
Table 8-1 – Specification of intra prediction mode and associated names	130
Table 8-2 – Specification of modeIdx	132
Table 8-3 – Specification of IntraPredModeC when ChromaArrayType is equal to 2.....	132
Table 8-4 – Specification of intraHorVerDistThres[nTbS] for various transform block sizes	137
Table 8-5 – Specification of intraPredAngle	139
Table 8-6 – Specification of invAngle.....	139
Table 8-7 – Specification of l0CandIdx and l1CandIdx	155

Table 8-8 – Assignment of the luma prediction sample <code>predSampleLX_L</code>	167
Table 8-9 – Assignment of the chroma prediction sample <code>predSampleLX_C</code> for (X, Y) being replaced by (1, b), (2, c), (3, d), (4, e), (5, f), (6, g) and (7, h), respectively	169
Table 8-10 – Specification of <code>Q_{pC}</code> as a function of <code>qPi</code> for <code>ChromaArrayType</code> equal to 1.....	176
Table 8-11 – Name of association to <code>edgeType</code>	185
Table 8-12 – Derivation of threshold variables β' and t_C' from input Q	194
Table 8-13 – Specification of <code>hPos</code> and <code>vPos</code> according to the sample adaptive offset class	200
Table 9-1 – Bit strings with "prefix" and "suffix" bits and assignment to <code>codeNum</code> ranges (informative)	201
Table 9-2 – Exp-Golomb bit strings and <code>codeNum</code> in explicit form and used as <code>ue(v)</code> (informative).....	202
Table 9-3 – Assignment of syntax element to <code>codeNum</code> for signed Exp-Golomb coded syntax elements <code>se(v)</code>	202
Table 9-4 – Association of <code>ctxIdx</code> and syntax elements for each <code>initializationType</code> in the initialization process	207
Table 9-5 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>sao_merge_left_flag</code> and <code>sao_merge_up_flag</code>	209
Table 9-6 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>sao_type_idx_luma</code> and <code>sao_type_idx_chroma</code>	209
Table 9-7 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>split_cu_flag</code>	209
Table 9-8 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>cu_transquant_bypass_flag</code>	209
Table 9-9 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>cu_skip_flag</code>	209
Table 9-10 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>pred_mode_flag</code>	209
Table 9-11 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>part_mode</code>	210
Table 9-12 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>prev_intra_luma_pred_flag</code>	210
Table 9-13 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>intra_chroma_pred_mode</code>	210
Table 9-14 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>rqt_root_cbf</code>	210
Table 9-15 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>merge_flag</code>	210
Table 9-16 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>merge_idx</code>	210
Table 9-17 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>inter_pred_idc</code>	211
Table 9-18 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>ref_idx_l0</code> and <code>ref_idx_l1</code>	211
Table 9-19 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>mvp_l0_flag</code> and <code>mvp_l1_flag</code>	211
Table 9-20 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>split_transform_flag</code>	211
Table 9-21 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>cbf_luma</code>	211
Table 9-22 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>cbf_cb</code> and <code>cbf_cr</code>	211
Table 9-23 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>abs_mvd_greater0_flag</code> and <code>abs_mvd_greater1_flag</code>	212
Table 9-24 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>cu_qp_delta_abs</code>	212
Table 9-25 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>transform_skip_flag</code>	212
Table 9-26 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>last_sig_coeff_x_prefix</code>	212
Table 9-27 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>last_sig_coeff_y_prefix</code>	212
Table 9-28 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>coded_sub_block_flag</code>	213
Table 9-29 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>sig_coeff_flag</code>	213
Table 9-30 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>coeff_abs_level_greater1_flag</code>	213
Table 9-31 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>coeff_abs_level_greater2_flag</code>	214
Table 9-32 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>explicit_rdpem_flag</code>	214
Table 9-33 – Values of <code>initValue</code> for <code>ctxIdx</code> of <code>explicit_rdpem_dir_flag</code>	214

Table 9-34 – Values of initValue for ctxIdx of cu_chroma_qp_offset_flag.....	214
Table 9-35 – Values of initValue for ctxIdx of cu_chroma_qp_offset_idx.....	214
Table 9-36 – Values of initValue for ctxIdx of log2_res_scale_abs_plus1	215
Table 9-37 – Values of initValue for ctxIdx of res_scale_sign_flag	215
Table 9-38 – Values of initValue for ctxIdx of palette_mode_flag.....	215
Table 9-39 – Values of initValue for ctxIdx of tu_residual_act_flag	215
Table 9-40 – Values of initValue for ctxIdx of palette_run_prefix	215
Table 9-41 – Values of initValue for ctxIdx of copy_above_palette_indices_flag and copy_above_indices_for_final_run_flag	216
Table 9-42 – Values of initValue for ctxIdx of palette_transpose_flag.....	216
Table 9-43 – Syntax elements and associated binarizations	218
Table 9-44 – Bin string of the unary binarization (informative).....	221
Table 9-45 – Binarization for part_mode.....	223
Table 9-46 – Binarization for intra_chroma_pred_mode.....	224
Table 9-47 – Binarization for inter_pred_idc	224
Table 9-48 – Assignment of ctxInc to syntax elements with context coded bins	228
Table 9-49 – Specification of ctxInc using left and above syntax elements	230
Table 9-50 – Specification of ctxIdxMap[i]	232
Table 9-51 – Specification of ctxIdxMap[copy_above_palette_indices_flag][binIdx]	234
Table 9-52 – Specification of rangeTabLps depending on the values of pStateIdx and qRangeIdx	238
Table 9-53 – State transition table	239
Table A.1 – Allowed values for syntax elements in the format range extensions profiles	253
Table A.2 – Bitstream indications for conformance to format range extensions profiles.....	254
Table A.3 – Bitstream indications for conformance to high throughput profiles	257
Table A.4 – Allowed values for syntax elements in the screen content coding extensions profiles	258
Table A.5 – Bitstream indications for conformance to screen content coding extensions profiles	259
Table A.6 – Allowed values for syntax elements in the high throughput screen content coding extensions profiles	261
Table A.7 – Bitstream indications for conformance to high throughput screen content coding extensions profiles	262
Table A.8 – General tier and level limits	264
Table A.9 – Tier and level limits for the video profiles.....	266
Table A.10 – Specification of CpbVclFactor, CpbNalFactor, FormatCapabilityFactor and MinCrScaleFactor.....	267
Table A.11 – Maximum picture rates (pictures per second) at level 1 to 4.1 for some example picture sizes when MinCbSizeY is equal to 64.....	269
Table A.12 – Maximum picture rates (pictures per second) at level 5 to 6.2 for some example picture sizes when MinCbSizeY is equal to 64.....	270
Table D.1 – Persistence scope of SEI messages (informative).....	318
Table D.2 – Interpretation of pic_struct.....	326
Table D.3 – scene_transition_type values	333
Table D.4 – film_grain_model_id values	336
Table D.5 – blending_mode_id values	337
Table D.6 – filter_hint_type values	342

Table D.7 – Interpretation of camera_iso_speed_idc and exposure_idx_idc	345
Table D.8 – Definition of frame_packing_arrangement_type	347
Table D.9 – Definition of content_interpretation_type.....	348
Table D.10 – Interpretation of hash_type	355
Table D.11 – Definition of counting_type[i] values	362
Table D.12 – Definition of segmented_rect_content_interpretation_type.....	366
Table D.13 – ver_chroma_filter_idc values.....	372
Table D.14 – hor_chroma_filter_idc values	373
Table D.15 – Chroma sampling format indicated by target_format_idc	373
Table D.16 – Constraints on the value of num_vertical_filters	374
Table D.17 – Constraints on the value of num_horizontal_filters	374
Table D.18 – Values of verFilterCoeff and verTapLength when ver_chroma_filter_idc is equal to 2.....	375
Table D.19 – Values of horFilterCoeff and horTapLength when hor_chroma_filter_idc is equal to 2.....	375
Table D.20 – Usage of chroma filter in the vertical direction	378
Table D.21 – Usage of chroma filter in the horizontal direction	380
Table D.22 – rwp_transform_type[i] values.....	396
Table D.23 – Interpretation of manifest_sei_description[i].....	411
Table E.1 – Interpretation of sample aspect ratio indicator	421
Table E.2 – Meaning of video_format.....	422
Table E.3 – Colour primaries interpretation using the colour_primaries syntax element.....	423
Table E.4 – Transfer characteristics interpretation using the transfer_characteristics syntax element	424
Table E.5 – Matrix coefficients interpretation using the matrix_coeffs syntax element.....	431
Table E.6 – Definition of HorizontalOffsetC and VerticalOffsetC as a function of chroma_format_idc and ChromaLocType.....	433
Table E.7 – Divisor for computation of DpbOutputElementalInterval[n]	439
Table F.1 – Mapping of ScalabiltyId to scalability dimensions.....	465
Table F.2 – Mapping of AuxId to the type of auxiliary pictures	467
Table F.3 – Specification of CompatibleProfileList	514
Table F.4 – Persistence scope of SEI messages (informative).....	537
Table G.1 – Persistence scope of SEI messages (informative).....	567
Table G.2 – Association between camera parameter variables and syntax elements	569
Table G.3 – Definition of depth_representation_type	571
Table G.4 – Association between depth parameter variables and syntax elements	571
Table G.5 – Association between camera parameter variables and syntax elements.	576
Table H.1 – 16-phase luma resampling filter.....	583
Table H.2 – 16-phase chroma resampling filter.....	584
Table H.3 – Allowed values for syntax elements in the scalable format range extensions profiles	596
Table H.4 – Bitstream indications for conformance to scalable range extensions profiles.....	597
Table I.1 – Name association to prediction mode and partitioning type.....	628
Table I.2 – Specification of intra prediction mode and associated names	635

Table I.3 – Specification of divCoeff depending on sDenomDiv.....	663
Table I.4 – Association of ctxIdx and syntax elements for each initializationType in the initialization process	676
Table I.5 – Values of initValue for skip_intra_flag ctxIdx.....	677
Table I.6 – Values of initValue for no_dim_flag ctxIdx.....	677
Table I.7 – Values of initValue for depth_intra_mode_idx_flag ctxIdx.....	677
Table I.8 – Values of initValue for skip_intra_mode_idx ctxIdx	677
Table I.9 – Values of initValue for dbbp_flag ctxIdx.....	677
Table I.10 – Values of initValue for dc_only_flag ctxIdx.....	677
Table I.11 – Values of initValue for iv_res_pred_weight_idx ctxIdx	677
Table I.12 – Values of initValue for illu_comp_flag ctxIdx.....	678
Table I.13 – Values of initValue for depth_dc_present_flag ctxIdx.....	678
Table I.14 – Values of initValue for depth_dc_abs ctxIdx	678
Table I.15 – Syntax elements and associated binarizations	679
Table I.16 – Binarization for part_mode.....	679
Table I.17 – Assignment of ctxInc to syntax elements with context coded bins	681
Table I.18 – Specification of ctxInc using left and above syntax elements	681
Table I.19 – Persistence scope of SEI messages (informative).....	686
Table I.20 – Interpretation of depth_type	687
Table I.21 – Locations of the top-left luma samples of constituent pictures packed in a picture with ViewIdx greater than 0 relative to the top-left luma sample of this picture	687

Foreword

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardizing telecommunications on a world-wide basis. The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups that, in turn, produce Recommendations on these topics. The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1. In some areas of information technology that fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for world-wide standardization. National Bodies that are members of ISO and IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

This Recommendation | International Standard was prepared jointly by ITU-T SG 16 Question 6/16, also known as VCEG (Video Coding Experts Group), and by the working groups of ISO/IEC JTC 1/SC 29 known as MPEG (Moving Picture Experts Group). VCEG was formed in 1997 to maintain prior ITU-T video coding standards and develop new video coding standard(s) appropriate for a wide range of conversational and non-conversational services. MPEG was formed in 1988 to establish standards for coding of moving pictures and associated audio for various applications such as digital storage media, distribution, and communication.

In this Recommendation | International Standard Annexes A through I contain normative requirements and are an integral part of this Recommendation | International Standard.

High efficiency video coding

0 Introduction

0.1 General

This clause and its subclauses do not form an integral part of this Recommendation | International Standard.

0.2 Prologue

As the costs for both processing power and memory have reduced, network support for coded video data has diversified, and advances in video coding technology have progressed, the need has arisen for an industry standard for compressed video representation with substantially increased coding efficiency and enhanced robustness to network environments. Toward these ends the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) formed a Joint Collaborative Team on Video Coding (JCT-VC) in 2010 and a Joint Collaborative Team on 3D Video Coding Extension Development (JCT-3V) in 2012 for development of a new Recommendation | International Standard. This Recommendation | International Standard was developed in the JCT-VC and the JCT-3V until 2020, when the responsibility for further maintenance and enhancement of the standard was transferred to another joint collaborative team of the same organizations called the Joint Video Experts Team (JVET).

0.3 Purpose

This Recommendation | International Standard was developed in response to the growing need for higher compression of moving pictures for various applications such as videoconferencing, digital storage media, television broadcasting, internet streaming, and communications. It is also designed to enable the use of the coded video representation in a flexible manner for a wide variety of network environments as well as to enable the use of multi-core parallel encoding and decoding devices. The use of this Recommendation | International Standard allows motion video to be manipulated as a form of computer data and to be stored on various storage media, transmitted and received over existing and future networks and distributed on existing and future broadcasting channels. Supports for higher bit depths and enhanced chroma formats, including the use of full-resolution chroma are provided. Support for scalability enables video transmission on networks with varying transmission conditions and other scenarios involving multiple bit rate services. Support for multiview enables representation of video content with multiple camera views and optional auxiliary information. Support for 3D enables joint representation of video content and depth information with multiple camera views.

0.4 Applications

This Recommendation | International Standard is designed to cover a broad range of applications for video content including but not limited to the following:

- Broadcast (cable TV on optical networks / copper, satellite, terrestrial, etc.)
- Camcorders
- Content production and distribution
- Digital cinema
- Home cinema
- Internet streaming, download and play
- Medical imaging
- Mobile streaming, broadcast and communications
- Real-time conversational services (videoconferencing, videophone, telepresence, etc.)
- Remote video surveillance
- Storage media (optical disks, digital video tape recorder, etc.)
- Wireless display

0.5 Publication and versions of this Specification

This Specification has been jointly developed by ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). It is published as technically-aligned twin text in both ITU-T and ISO/IEC. As the basis text has been drafted to become both an ITU-T Recommendation and an ISO/IEC International Standard, the term "Specification" (with capitalization to indicate that it refers to the whole of the text) is used herein when the text refers to itself.

Rec. ITU-T H.265 | ISO/IEC 23008-2 version 1 refers to the first approved version of this Recommendation | International Standard. The first edition published by ITU-T as Rec. ITU-T H.265 (04/2013) and by ISO/IEC as ISO/IEC 23008-2:2013 corresponded to the first version.

Rec. ITU-T H.265 | ISO/IEC 23008-2 version 2 refers to the integrated text additionally containing format range extensions, scalability extensions, multiview extensions, additional supplement enhancement information, and corrections to various minor defects in the prior content of the Specification. The second edition published by ITU-T as Rec. H.265 (10/2014) and by ISO/IEC as ISO/IEC 23008-2:2015 corresponded to the second version.

Rec. ITU-T H.265 | ISO/IEC 23008-2 version 3 refers to the integrated text additionally containing 3D extensions, additional supplement enhancement information, and corrections to various minor defects in the prior content of the Specification. The third edition published by ITU-T as Rec. H.265 (04/2015) corresponded to the third version.

Rec. ITU-T H.265 | ISO/IEC 23008-2 version 4 refers to the integrated text additionally containing screen content coding extensions profiles, scalable range extensions profiles, additional high throughput profiles, additional supplement enhancement information, additional colour representation identifiers, and corrections to various minor defects in the prior content of the Specification. The fourth edition published by ITU-T as Rec. H.265 (12/2016) and by ISO/IEC as ISO/IEC 23008-2:2017 corresponded to the fourth version.

Rec. ITU-T H.265 | ISO/IEC 23008-2 version 5 refers to the integrated text additionally containing additional SEI messages that include omnidirectional video SEI messages, a Monochrome 10 profile, a Main 10 Still Picture profile, and corrections to various minor defects in the prior content of the Specification. The fifth edition published by ITU-T as Rec. H.265 (02/2018) corresponded to the fifth version.

Rec. ITU-T H.265 | ISO/IEC 23008-2 version 6 refers to the integrated text additionally containing additional SEI messages for SEI manifest and SEI prefix, along with some corrections to the existing specification text. The sixth edition published by ITU-T as Rec. H.265 (06/2019) corresponded to the sixth version.

Rec. ITU-T H.265 | ISO/IEC 23008-2 version 7 refers to the integrated text additionally containing the fisheye video information SEI message and the annotated regions SEI message, along with some corrections to the existing specification text. The seventh edition published by ITU-T as Rec. H.265 (11/2019) corresponded to the seventh version.

Rec. ITU-T H.265 | ISO/IEC 23008-2 version 8 (the current version) refers to the integrated text additionally containing the shutter interval information SEI message, along with some corrections to the existing specification text.

0.6 Profiles, tiers and levels

This Recommendation | International Standard is designed to be generic in the sense that it serves a wide range of applications, bit rates, resolutions, qualities and services. Applications should cover, among other things, digital storage media, television broadcasting and real-time communications. In the course of creating this Specification, various requirements from typical applications have been considered, necessary algorithmic elements have been developed, and these have been integrated into a single syntax. Hence, this Specification will facilitate video data interchange among different applications.

Considering the practicality of implementing the full syntax of this Specification, however, a limited number of subsets of the syntax are also stipulated by means of "profiles", "tiers" and "levels". These and other related terms are formally defined in clause 3.

A "profile" is a subset of the entire bitstream syntax that is specified in this Recommendation | International Standard. Within the bounds imposed by the syntax of a given profile, it is still possible to require a very large variation in the performance of encoders and decoders depending upon the values taken by syntax elements in the bitstream such as the specified size of the decoded pictures. In many applications, it is currently neither practical nor economical to implement a decoder capable of dealing with all hypothetical uses of the syntax within a particular profile.

In order to deal with this problem, "tiers" and "levels" are specified within each profile. A level of a tier is a specified set of constraints imposed on values of the syntax elements in the bitstream. These constraints may be simple limits on values. Alternatively they may take the form of constraints on arithmetic combinations of values (e.g., picture width multiplied by picture height multiplied by number of pictures decoded per second). A level specified for a lower tier is more constrained than a level specified for a higher tier.

Coded video content conforming to this Recommendation | International Standard uses a common syntax. In order to achieve a subset of the complete syntax, flags, parameters and other syntax elements are included in the bitstream that signal the presence or absence of syntactic elements that occur later in the bitstream.

0.7 Overview of the design characteristics

The coded representation specified in the syntax is designed to enable a high compression capability for a desired image or video quality. The algorithm is typically not lossless, as the exact source sample values are typically not preserved through the encoding and decoding processes. A number of techniques may be used to achieve highly efficient compression. Encoding algorithms (not specified in this Recommendation | International Standard) may select between inter and intra coding for block-shaped regions of each picture. Inter coding uses motion vectors for block-based inter prediction to exploit temporal statistical dependencies between different pictures. Intra coding uses various spatial prediction modes to exploit spatial statistical dependencies in the source signal for a single picture. Motion vectors and intra prediction modes may be specified for a variety of block sizes in the picture. The prediction residual may then be further compressed using a transform to remove spatial correlation inside the transform block before it is quantized, producing a possibly irreversible process that typically discards less important visual information while forming a close approximation to the source samples. Finally, the motion vectors or intra prediction modes may also be further compressed using a variety of prediction mechanisms, and, after prediction, are combined with the quantized transform coefficient information and encoded using arithmetic coding.

0.8 How to read this Specification

It is suggested that the reader starts with clause 1 (Scope) and moves on to clause 3 (Definitions). Clause 6 should be read for the geometrical relationship of the source, input and output of the decoder. Clause 7 (Syntax and semantics) specifies the order to parse syntax elements from the bitstream. See clauses 7.1–7.3 for syntactical order and see clause 7.4 for semantics; e.g., the scope, restrictions and conditions that are imposed on the syntax elements. The actual parsing for most syntax elements is specified in clause 9 (Parsing process). Clause 10 (Sub-bitstream extraction process) specifies the sub-bitstream extraction process. Finally, clause 8 (Decoding process) specifies how the syntax elements are mapped into decoded samples. Throughout reading this Specification, the reader should refer to clauses 2 (Normative references), 4 (Abbreviations), and 5 (Conventions) as needed. Annexes A through I also form an integral part of this Recommendation | International Standard.

Annex A specifies profiles each being tailored to certain application domains, and defines the so-called tiers and levels of the profiles. Annex B specifies syntax and semantics of a byte stream format for delivery of coded video as an ordered stream of bytes. Annex C specifies the hypothetical reference decoder, bitstream conformance, decoder conformance and the use of the hypothetical reference decoder to check bitstream and decoder conformance. Annex D specifies syntax and semantics for supplemental enhancement information message payloads. Annex E specifies syntax and semantics of the video usability information parameters of the sequence parameter set. Annex F specifies general multi-layer support for bitstreams and decoders. Annex G contains support for multiview coding. Annex H contains support for scalability. Annex I contains support for 3D coding.

Throughout this Specification, statements appearing with the preamble "NOTE –" are informative and are not an integral part of this Recommendation | International Standard.

1 Scope

This Recommendation | International Standard specifies high efficiency video coding.

2 Normative references

2.1 General

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

2.2 Identical Recommendations | International Standards

- None

2.3 Paired Recommendations | International Standards equivalent in technical content

- None

2.4 Additional references

- Recommendation ITU-T T.35 (in force), *Procedure for the allocation of ITU-T defined codes for non-standard facilities*.
- ISO/IEC 10646: in force, *Information technology – Universal Coded Character Set (UCS)*.
- ISO/IEC 11578: in force, *Information technology – Open Systems Interconnection – Remote Procedure Call (RPC)*.
- ISO 11664-1: in force, *Colorimetry – Part 1: CIE standard colorimetric observers*.
- ISO 12232: in force, *Photography – Digital still cameras – Determination of exposure index, ISO speed ratings, standard output sensitivity, and recommended exposure index*.
- IETF RFC 1321 (in force), *The MD5 Message-Digest Algorithm*.
- IETF RFC 5646 (in force), *Tags for Identifying Languages*.
- ISO/IEC 23001-11 (in force), *Information Technology – MPEG Systems technologies – Part 11: Energy-efficient media consumption (green metadata)*.

3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply:

- 3.1 access unit:** A set of *NAL units* that are associated with each other according to a specified classification rule, are consecutive in *decoding order*, and contain exactly one *coded picture* with *nuh_layer_id* equal to 0.
- NOTE 1 – In addition to containing the video coding layer (VCL) *NAL units* of the coded picture with *nuh_layer_id* equal to 0, an access unit may also contain non-VCL *NAL units*. The decoding of an access unit with the decoding process specified in clause 8 always results in a decoded picture with *nuh_layer_id* equal to 0.
- NOTE 2 – An access unit is defined differently in Annex F and does not need to contain a coded picture with *nuh_layer_id* equal to 0.
- 3.2 AC transform coefficient:** Any *transform coefficient* for which the *frequency index* in at least one of the two dimensions is non-zero.
- 3.3 associated IRAP picture:** The previous *IRAP picture* in *decoding order* (when present).
- 3.4 associated non-VCL NAL unit:** A *non-VCL NAL unit* (when present) for a *VCL NAL unit* where the *VCL NAL unit* is the *associated VCL NAL unit* of the *non-VCL NAL unit*.
- 3.5 associated VCL NAL unit:** The preceding *VCL NAL unit* in *decoding order* for a *non-VCL NAL unit* with *nal_unit_type* equal to *EOS_NUT*, *EOB_NUT*, *FD_NUT* or *SUFFIX_SEI_NUT*, or in the ranges of *RSV_NVCL45*..*RSV_NVCL47* or *UNSPEC56*..*UNSPEC63*; or otherwise the next *VCL NAL unit* in *decoding order*.
- 3.6 azimuth circle:** A circle on a sphere connecting all points with the same azimuth value.
- NOTE – An azimuth circle is always a *great circle* like a longitude line on the earth.
- 3.7 base layer:** A *layer* in which all *NAL units* have *nuh_layer_id* equal to 0.
- 3.8 bin:** One bit of a *bin string*.
- 3.9 binarization:** A set of *bin strings* for all possible values of a *syntax element*.
- 3.10 binarization process:** A unique mapping process of all possible values of a *syntax element* onto a set of *bin strings*.
- 3.11 bin string:** An intermediate binary representation of values of *syntax elements* from the *binarization* of the *syntax element*.
- 3.12 bi-predictive (B) slice:** A *slice* that is decoded using *intra prediction* or using *inter prediction* with at most two *motion vectors* and *reference indices* to *predict* the sample values of each *block*.

- 3.13 bitstream:** A sequence of bits, in the form of a *NAL unit stream* or a *byte stream*, that forms the representation of *coded pictures* and associated data forming one or more coded video sequences (*CVSs*).
- 3.14 block:** An MxN (M-column by N-row) array of samples, or an MxN array of *transform coefficients*.
- 3.15 broken link:** A location in a *bitstream* at which it is indicated that some subsequent *pictures* in *decoding order* may contain serious visual artefacts due to unspecified operations performed in the generation of the *bitstream*.
- 3.16 broken link access (BLA) access unit:** An *access unit* in which the *coded picture* with *nuh_layer_id* equal to 0 is a *BLA picture*.
- 3.17 broken link access (BLA) picture:** An *IRAP picture* for which each *VCL NAL unit* has *nal_unit_type* equal to BLA_W_LP, BLA_W_RADL, or BLA_N_LP.
- NOTE – A BLA picture does not refer to any pictures other than itself for inter prediction in its decoding process, and may be the first picture in the bitstream in decoding order, or may appear later in the bitstream. Each BLA picture begins a new CVS, and has the same effect on the decoding process as an instantaneous decoding refresh (IDR) picture. However, a BLA picture contains syntax elements that specify a non-empty RPS. When a BLA picture for which each VCL NAL unit has *nal_unit_type* equal to BLA_W_LP, it may have associated random access skipped leading (RASL) pictures, which are not output by the decoder and may not be decodable, as they may contain references to pictures that are not present in the bitstream. When a BLA picture for which each VCL NAL unit has *nal_unit_type* equal to BLA_W_LP, it may also have associated RADL pictures, which are specified to be decoded. When a BLA picture for which each VCL NAL unit has *nal_unit_type* equal to BLA_W_RADL, it does not have associated RASL pictures but may have associated random access decodable leading (RADL) pictures. When a BLA picture for which each VCL NAL unit has *nal_unit_type* equal to BLA_N_LP, it does not have any associated leading pictures.
- 3.18 buffering period:** The set of *access units* starting with an *access unit* that contains a buffering period supplemental enhancement information (SEI) message and containing all subsequent *access units* in *decoding order* up to but not including the next *access unit* (when present) that contains a buffering period SEI message.
- 3.19 byte:** A sequence of 8 bits, within which, when written or read as a sequence of bit values, the left-most and right-most bits represent the most and least significant bits, respectively.
- 3.20 byte-aligned:** A position in a *bitstream* is byte-aligned when the position is an integer multiple of 8 bits from the position of the first bit in the *bitstream*, and a bit or *byte* or *syntax element* is said to be byte-aligned when the position at which it appears in a *bitstream* is byte-aligned.
- 3.21 byte stream:** An encapsulation of a *NAL unit stream* containing *start code prefixes* and *NAL units* as specified in Annex B.
- 3.22 can:** A term used to refer to behaviour that is allowed, but not necessarily required.
- 3.23 chroma:** An adjective, represented by the symbols Cb and Cr, specifying that a sample array or single sample is representing one of the two colour difference signals related to the primary colours.
- NOTE – The term chroma is used rather than the term chrominance in order to avoid the implication of the use of linear light transfer characteristics that is often associated with the term chrominance.
- 3.24 clean random access (CRA) access unit:** An *access unit* in which the *coded picture* with *nuh_layer_id* equal to 0 is a *CRA picture*.
- 3.25 clean random access (CRA) picture:** An *IRAP picture* for which each *VCL NAL unit* has *nal_unit_type* equal to CRA_NUT.
- NOTE – A CRA picture does not refer to any pictures other than itself for inter prediction in its decoding process, and may be the first picture in the bitstream in decoding order, or may appear later in the bitstream. A CRA picture may have associated RADL or RASL pictures. As with a BLA picture, a CRA picture may contain syntax elements that specify a non-empty RPS. When a CRA picture has *NoRaslOutputFlag* equal to 1, the associated RASL pictures are not output by the decoder, because they may not be decodable, as they may contain references to pictures that are not present in the bitstream.
- 3.26 coded picture:** A *coded representation* of a *picture* containing all *CTUs* of the *picture*.
- 3.27 coded picture buffer (CPB):** A first-in first-out buffer containing *decoding units* in *decoding order* specified in the *hypothetical reference decoder* in Annex C.
- 3.28 coded representation:** A data element as represented in its coded form.
- 3.29 coded slice segment NAL unit:** A *NAL unit* that has *nal_unit_type* in the range of TRAIL_N to RASL_R, inclusive, or in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive, which indicates that the *NAL unit* contains a coded *slice segment*.
- 3.30 coded layer-wise video sequence (CLVS):** A sequence of *pictures* and the associated non-VCL *NAL units* of the *base layer* of a *coded video sequence (CVS)*.

- 3.31 coded video sequence (CVS):** A sequence of *access units* that consists, in *decoding order*, of an *IRAP access unit* with NoRasOutputFlag equal to 1, followed by zero or more *access units* that are not *IRAP access units* with NoRasOutputFlag equal to 1, including all subsequent *access units* up to but not including any subsequent *access unit* that is an *IRAP access unit* with NoRasOutputFlag equal to 1.
- NOTE – An *IRAP access unit* may be an *IDR access unit*, a *BLA access unit*, or a *CRA access unit*. The value of NoRasOutputFlag is equal to 1 for each *IDR access unit*, each *BLA access unit*, and each *CRA access unit* that is the first access unit in the bitstream in decoding order, is the first access unit that follows an end of sequence NAL unit in decoding order, or has HandleCraAsBlaFlag equal to 1.
- 3.32 coded video sequence group (CVSG):** One or more consecutive *CVSs* in *decoding order* that collectively consist of an *IRAP access unit* that activates a video parameter set (VPS) RBSP firstVpsRbsp that was not already active followed by all subsequent *access units*, in *decoding order*, for which firstVpsRbsp is the active VPS raw byte sequence payload (RBSP) up to the end of the *bitstream* or up to but excluding the *access unit* that activates a different VPS RBSP than firstVpsRbsp, whichever is earlier in *decoding order*.
- 3.33 coding block:** An $N \times N$ *block* of samples for some value of N such that the division of a *CTB* into *coding blocks* is a *partitioning*.
- 3.34 coding tree block (CTB):** An $N \times N$ *block* of samples for some value of N such that the division of a *component* into *CTBs* is a *partitioning*.
- 3.35 coding tree unit (CTU):** A *CTB* of *luma* samples, two corresponding *CTBs* of *chroma* samples of a *picture* that has three sample arrays, or a *CTB* of samples of a monochrome *picture* or a *picture* that is coded using three separate colour planes and *syntax structures* used to code the samples.
- 3.36 coding unit:** A *coding block* of *luma* samples, two corresponding *coding blocks* of *chroma* samples of a *picture* that has three sample arrays, or a *coding block* of samples of a monochrome *picture* or a *picture* that is coded using three separate colour planes and *syntax structures* used to code the samples.
- 3.37 component:** An array or single sample from one of the three arrays (*luma* and two *chroma*) that compose a *picture* in 4:2:0, 4:2:2, or 4:4:4 colour format or the array or a single sample of the array that compose a *picture* in monochrome format.
- 3.38 constituent picture:** A part of a spatially *frame*-packed stereoscopic *picture* that corresponds to one view, or a *picture* itself when *frame* packing is not in use or the temporal interleaving *frame* packing arrangement is in use.
- 3.39 context variable:** A variable specified for the *adaptive binary arithmetic decoding process* of a *bin* by an equation containing recently decoded *bins*.
- 3.40 cropped decoded picture:** The result of cropping a *decoded picture* based on the conformance cropping window specified in the *sequence parameter set (SPS)* that is referred to by the corresponding *coded picture*.
- 3.41 decoded picture:** A *decoded picture* is derived by decoding a *coded picture*.
- 3.42 decoded picture buffer (DPB):** A buffer holding *decoded pictures* for reference, output reordering, or output delay specified for the *hypothetical reference decoder* in Annex C.
- 3.43 decoder:** An embodiment of a *decoding process*.
- 3.44 decoder under test (DUT):** A *decoder* that is tested for conformance to this Specification by operating the *hypothetical stream scheduler* to deliver a conforming *bitstream* to the *decoder* and to the *hypothetical reference decoder* and comparing the values and timing or order of the output of the two *decoders*.
- 3.45 decoding order:** The order in which *syntax elements* are processed by the *decoding process*.
- 3.46 decoding process:** The process specified in this Specification that reads a *bitstream* and derives *decoded pictures* from it.
- 3.47 decoding unit:** An *access unit* if SubPicHrdFlag is equal to 0 or a subset of an *access unit* otherwise, consisting of one or more *VCL NAL units* in an *access unit* and the *associated non-VCL NAL units*.
- 3.48 dependent slice segment:** A *slice segment* for which the values of some *syntax elements* of the *slice segment header* are inferred from the values for the preceding *independent slice segment* in *decoding order*.
- 3.49 display process:** A process not specified in this Specification having, as its input, the *cropped decoded pictures* that are the output of the *decoding process*.
- 3.50 elementary stream:** A sequence of one or more *bitstreams*.
- NOTE – An elementary stream that consists of two or more *bitstreams* would typically have been formed by splicing together two or more *bitstreams* (or parts thereof).

- 3.51 elevation circle:** A circle on a sphere connecting all points with the same elevation value.
- NOTE – An elevation circle is similar to a latitude line on the earth. Except when the elevation value is zero, an elevation circle is not a *great circle* like a longitude circle on the earth.
- 3.52 emulation prevention byte:** A byte equal to 0x03 that is present within a *NAL unit* when the *syntax elements* of the *bitstream* form certain patterns of *byte* values in a manner that ensures that no sequence of consecutive *byte-aligned bytes* in the *NAL unit* can contain a *start code prefix*.
- 3.53 encoder:** An embodiment of an *encoding process*.
- 3.54 encoding process:** A process not specified in this Specification that produces a *bitstream* conforming to this Specification.
- 3.55 field:** An assembly of alternative rows of samples of a *frame*.
- 3.56 filler data NAL units:** *NAL units* with *nal_unit_type* equal to FD_NUT.
- 3.57 flag:** A variable or single-bit *syntax element* that can take one of the two possible values: 0 and 1.
- 3.58 frame:** The composition of a top *field* and a bottom *field*, where sample rows 0, 2, 4, ... originate from the top *field* and sample rows 1, 3, 5, ... originate from the bottom *field*.
- 3.59 frequency index:** A one-dimensional or two-dimensional index associated with a *transform coefficient* prior to the application of a *transform* in the *decoding process*.
- 3.60 global coordinate axes:** The coordinate axes associated with *omnidirectional video* that are associated with an externally referenceable position and orientation.
- NOTE – The global coordinate axes may correspond to the position and orientation of a device or rig used for omnidirectional audio/video acquisition as well as the position of an observer's head in the three-dimensional space of the *omnidirectional video* rendering environment.
- 3.61 great circle:** The intersection of a sphere and a plane that passes through the centre point of the sphere.
- NOTE – A great circle is also known as an orthodrome or Riemannian circle.
- 3.62 hypothetical reference decoder (HRD):** A hypothetical *decoder* model that specifies constraints on the variability of conforming *NAL unit streams* or conforming *byte streams* that an encoding process may produce.
- 3.63 hypothetical stream scheduler (HSS):** A hypothetical delivery mechanism used for checking the conformance of a *bitstream* or a *decoder* with regards to the timing and data flow of the input of a *bitstream* into the *hypothetical reference decoder*.
- 3.64 independent slice segment:** A *slice segment* for which the values of the *syntax elements* of the *slice segment header* are not inferred from the values for a preceding *slice segment*.
- 3.65 informative:** A term used to refer to content provided in this Specification that does not establish any mandatory requirements for conformance to this Specification and thus is not considered an integral part of this Specification.
- 3.66 instantaneous decoding refresh (IDR) access unit:** An *access unit* in which the *coded picture* with *nuh_layer_id* equal to 0 is an *IDR picture*.
- 3.67 instantaneous decoding refresh (IDR) picture:** An *IRAP picture* for which each *VCL NAL unit* has *nal_unit_type* equal to IDR_W_RADL or IDR_N_LP.
- NOTE – An IDR picture does not refer to any pictures other than itself for inter prediction in its decoding process, and may be the first picture in the bitstream in decoding order, or may appear later in the bitstream. Each IDR picture is the first picture of a CVS in decoding order. When an IDR picture for which each VCL NAL unit has *nal_unit_type* equal to IDR_W_RADL, it may have associated RADL pictures. When an IDR picture for which each VCL NAL unit has *nal_unit_type* equal to IDR_N_LP, it does not have any associated leading pictures. An IDR picture does not have associated RASL pictures.
- 3.68 inter coding:** Coding of a *coding block*, *slice*, or *picture* that uses *inter prediction*.
- 3.69 inter prediction:** A *prediction* derived in a manner that is dependent on data elements (e.g., sample values or motion vectors) of one or more *reference pictures*.
- NOTE – A prediction from a reference picture that is the current picture itself is also inter prediction.
- 3.70 intra coding:** Coding of a *coding block*, *slice*, or *picture* that uses *intra prediction*.
- 3.71 intra prediction:** A *prediction* derived from only data elements (e.g., sample values) of the same decoded *slice* without referring to a *reference picture*.

- 3.72 intra random access point (IRAP) access unit:** An *access unit* in which the *coded picture* with *nuh_layer_id* equal to 0 is an *IRAP picture*.
- 3.73 intra random access point (IRAP) picture:** A *coded picture* for which each *VCL NAL unit* has *nal_unit_type* in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive.
- NOTE – An IRAP picture does not refer to any pictures other than itself for inter prediction in its decoding process, and may be a BLA picture, a CRA picture or an IDR picture. The first picture in the bitstream in decoding order must be an IRAP picture. Provided the necessary parameter sets are available when they need to be activated, the IRAP picture and all subsequent non-RASL pictures in decoding order can be correctly decoded without performing the decoding process of any pictures that precede the IRAP picture in decoding order. There may be pictures in a bitstream that do not refer to any pictures other than itself for inter prediction in its decoding process that are not IRAP pictures.
- 3.74 intra (I) slice:** A *slice* that is decoded using *intra prediction* only.
- 3.75 layer:** A set of *VCL NAL units* that all have a particular value of *nuh_layer_id* and the *associated non-VCL NAL units*, or one of a set of syntactical structures having a hierarchical relationship.
- NOTE – Depending on the context, either the first layer concept or the second layer concept applies. The first layer concept is also referred to as a scalable layer, wherein a layer may be a spatial scalable layer, a quality scalable layer, a view, etc. A temporal true subset of a scalable layer is not referred to as a layer but referred to as a sub-layer or temporal sub-layer. The second layer concept is also referred to as a coding layer, wherein higher layers contain lower layers, and the coding layers are the CVS, picture, slice, slice segment, and CTU layers.
- 3.76 layer identifier list:** A list of *nuh_layer_id* values that is associated with a *layer set* or an *operation point* and can be used as an input to the *sub-bitstream extraction process*.
- 3.77 layer set:** A set of *layers* represented within a *bitstream* created from another *bitstream* by operation of the *sub-bitstream extraction process* with the another *bitstream*, the target highest TemporalId equal to 6, and the target *layer identifier list* equal to the *layer identifier list* associated with the layer set as inputs.
- 3.78 leading picture:** A *picture* that precedes the *associated IRAP picture* in *output order*.
- 3.79 leaf:** A terminating node of a tree that is a root node of a tree of depth 0.
- 3.80 level:** A defined set of constraints on the values that may be taken by the *syntax elements* and variables of this Specification, or the value of a *transform coefficient* prior to *scaling*.
- NOTE – The same set of levels is defined for all profiles, with most aspects of the definition of each level being in common across different profiles. Individual implementations may, within the specified constraints, support a different level for each supported profile.
- 3.81 list 0 (list 1) motion vector:** A *motion vector* associated with a *reference index* pointing into *reference picture list 0 (list 1)*.
- 3.82 list 0 (list 1) prediction:** *Inter prediction* of the content of a *slice* using a *reference index* pointing into *reference picture list 0 (list 1)*.
- 3.83 local coordinate axes:** The coordinate axes having a specified rotation relationship relative to the *global coordinate axes*.
- 3.84 long-term reference picture:** A *picture* that is marked as "used for long-term reference".
- 3.85 long-term reference picture set:** The two reference picture set (RPS) lists that may contain long-term reference pictures.
- 3.86 luma:** An adjective, represented by the symbol or subscript Y or L, specifying that a sample array or single sample is representing the monochrome signal related to the primary colours.
- NOTE – The term luma is used rather than the term luminance in order to avoid the implication of the use of linear light transfer characteristics that is often associated with the term luminance. The symbol L is sometimes used instead of the symbol Y to avoid confusion with the symbol y as used for vertical location.
- 3.87 may:** A term that is used to refer to behaviour that is allowed, but not necessarily required.
- NOTE – In some places where the optional nature of the described behaviour is intended to be emphasized, the phrase "may or may not" is used to provide emphasis.
- 3.88 motion vector:** A two-dimensional vector used for *inter prediction* that provides an offset from the coordinates in the *decoded picture* to the coordinates in a *reference picture*.
- 3.89 must:** A term that is used in expressing an observation about a requirement or an implication of a requirement that is specified elsewhere in this Specification (used exclusively in an *informative* context).
- 3.90 network abstraction layer (NAL) unit:** A *syntax structure* containing an indication of the type of data to follow and *bytes* containing that data in the form of an *RBSP* interspersed as necessary with *emulation prevention bytes*.

- 3.91 network abstraction layer (NAL) unit stream:** A sequence of *NAL units*.
- 3.92 non-scalable-nested SEI message:** An SEI message that is not contained in a scalable nesting SEI message.
- 3.93 non-reference picture:** A *picture* that is marked as "unused for reference".
 NOTE – A non-reference picture contains samples that cannot be used for inter prediction in the decoding process of subsequent pictures in decoding order. In other words, once a picture is marked as "unused for reference", it can never be marked back as "used for reference".
- 3.94 non-VCL NAL unit:** A *NAL unit* that is not a *VCL NAL unit*.
- 3.95 note:** A term that is used to prefix *informative* remarks (used exclusively in an *informative* context).
- 3.96 omnidirectional video:** A video content in a format that enables rendering according to the user's viewing orientation, e.g., if viewed using a head-mounted device, or according to a user's desired *viewport*, reflecting a potentially rotated viewing position.
- 3.97 operation point:** A *bitstream* created from another *bitstream* by operation of the *sub-bitstream extraction process* with the another *bitstream*, a target highest TemporalId, and a target *layer identifier list* as inputs.
 NOTE – If the target highest TemporalId of an operation point is equal to the greatest value of TemporalId in the layer set associated with the target layer identification list, the operation point is identical to the layer set. Otherwise it is a subset of the layer set.
- 3.98 output order:** The order in which the *decoded pictures* are output from the *decoded picture buffer* (for the *decoded pictures* that are to be output from the *decoded picture buffer*).
- 3.99 output time:** A time when a *decoded picture* is to be output as specified by the *hypothetical reference decoder (HRD)* according to the output timing *decoded picture buffer (DPB)* operation.
- 3.100 packed region:** A region in a *region-wise packed picture* that is mapped to a *projected region* according to a *region-wise packing*.
- 3.101 parameter:** A *syntax element* of a *video parameter set (VPS)*, *sequence parameter set (SPS)* or *picture parameter set (PPS)*, or the second word of the defined term *quantization parameter*.
- 3.102 partitioning:** The division of a set into subsets such that each element of the set is in exactly one of the subsets.
- 3.103 picture:** An array of *luma* samples in monochrome format or an array of *luma* samples and two corresponding arrays of *chroma* samples in 4:2:0, 4:2:2, and 4:4:4 colour format.
 NOTE – A picture may be either a frame or a field. However, in one CVS, either all pictures are frames or all pictures are fields.
- 3.104 picture parameter set (PPS):** A *syntax structure* containing *syntax elements* that apply to zero or more entire *coded pictures* as determined by a *syntax element* found in each *slice segment header*.
- 3.105 picture order count (POC):** A variable that is associated with each *picture*, uniquely identifies the associated *picture* among all *pictures* in the CVS, and, when the associated *picture* is to be output from the *decoded picture buffer*, indicates the position of the associated *picture* in *output order* relative to the *output order* positions of the other *pictures* in the same CVS that are to be output from the *decoded picture buffer*.
- 3.106 picture unit:** A set of *NAL units* that contain all *VCL NAL units* of a *coded picture* and their *associated non-VCL NAL units*.
- 3.107 prediction:** An embodiment of the *prediction process*.
- 3.108 prediction block:** A rectangular MxN *block* of samples on which the same *prediction* is applied.
- 3.109 prediction process:** The use of a *predictor* to provide an estimate of the data element (e.g., sample value or motion vector) currently being decoded.
- 3.110 prediction unit:** A *prediction block* of *luma* samples, two corresponding *prediction blocks* of *chroma* samples of a *picture* that has three sample arrays, or a *prediction block* of samples of a monochrome *picture* or a *picture* that is coded using three separate colour planes and *syntax structures* used to predict the *prediction block* samples.
- 3.111 predictive (P) slice:** A *slice* that is decoded using *intra prediction* or using *inter prediction* with at most one *motion vector* and *reference index* to *predict* the sample values of each *block*.
- 3.112 predictor:** A combination of specified values or previously decoded data elements (e.g., sample value or motion vector) used in the *decoding process* of subsequent data elements.
- 3.113 prefix SEI message:** An SEI message that is contained in a *prefix SEI NAL unit*.
- 3.114 prefix SEI NAL unit:** An *SEI NAL unit* that has *nal_unit_type* equal to PREFIX_SEI_NUT.

- 3.115 profile:** A specified subset of the syntax of this Specification.
- 3.116 projected picture:** A picture that uses a *projection* format for *omnidirectional video*.
- 3.117 projected region:** A region in a *projected picture* that is mapped to a *packed region* according to a *region-wise packing*.
- 3.118 projection:** A specified correspondence between the colour samples of a *projected picture* and azimuth and elevation positions on a sphere.
- 3.119 pulse code modulation (PCM):** Coding of the samples of a *block* by directly representing the sample values without *prediction* or application of a transform.
- 3.120 quadtree:** A *tree* in which a parent node can be split into four child nodes, each of which may become parent node for another split into four child nodes.
- 3.121 quantization parameter:** A variable used by the *decoding process* for *scaling of transform coefficient levels*.
- 3.122 random access:** The act of starting the decoding process for a *bitstream* at a point other than the beginning of the stream.
- 3.123 random access decodable leading (RADL) access unit:** An *access unit* in which the *coded picture* with *nuh_layer_id* equal to 0 is a *RADL picture*.
- 3.124 random access decodable leading (RADL) picture:** A *coded picture* for which each *VCL NAL unit* has *nal_unit_type* equal to RADL_R or RADL_N.
 NOTE – All RADL pictures are leading pictures. RADL pictures are not used as reference pictures for the decoding process of trailing pictures of the same associated IRAP picture. When present, all RADL pictures precede, in decoding order, all trailing pictures of the same associated IRAP picture.
- 3.125 random access skipped leading (RASL) access unit:** An *access unit* in which the *coded picture* with *nuh_layer_id* equal to 0 is a *RASL picture*.
- 3.126 random access skipped leading (RASL) picture:** A *coded picture* for which each *VCL NAL unit* has *nal_unit_type* equal to RASL_R or RASL_N.
 NOTE – All RASL pictures are leading pictures of an associated BLA or CRA picture. When the associated IRAP picture has *NoRaslOutputFlag* equal to 1, the RASL picture is not output and may not be correctly decodable, as the RASL picture may contain references to pictures that are not present in the bitstream. RASL pictures are not used as reference pictures for the decoding process of non-RASL pictures. When present, all RASL pictures precede, in decoding order, all trailing pictures of the same associated IRAP picture.
- 3.127 raster scan:** A mapping of a rectangular two-dimensional pattern to a one-dimensional pattern such that the first entries in the one-dimensional pattern are from the first top row of the two-dimensional pattern scanned from left to right, followed similarly by the second, third, etc., rows of the pattern (going down) each scanned from left to right.
- 3.128 raw byte sequence payload (RBSP):** A *syntax structure* containing an integer number of *bytes* that is encapsulated in a *NAL unit* and that is either empty or has the form of a *string of data bits* containing *syntax elements* followed by an *RBSP stop bit* and zero or more subsequent bits equal to 0.
- 3.129 raw byte sequence payload (RBSP) stop bit:** A bit equal to 1 present within a *raw byte sequence payload (RBSP)* after a *string of data bits*, for which the location of the end within an *RBSP* can be identified by searching from the end of the *RBSP* for the *RBSP stop bit*, which is the last non-zero bit in the *RBSP*.
- 3.130 recovery point:** A point in the *bitstream* at which the recovery of an exact or an approximate representation of the *decoded pictures* represented by the *bitstream* is achieved after a *random access* or *broken link*.
- 3.131 reference index:** An index into a *reference picture list*.
- 3.132 reference picture:** A *picture* that is a *short-term reference picture* or a *long-term reference picture*.
 NOTE – A reference picture contains samples that may be used for inter prediction in the decoding process of subsequent pictures in decoding order.
- 3.133 reference picture list:** A list of *reference pictures* that is used for *inter prediction* of a *P* or *B slice*.
 NOTE – For the decoding process of a *P* slice, there is one reference picture list – reference picture list 0. For the decoding process of a *B* slice, there are two reference picture lists – reference picture list 0 and reference picture list 1.
- 3.134 reference picture list 0:** The *reference picture list* used for *inter prediction* of a *P* or the first *reference picture list* used for *inter prediction* of a *B slice*.
- 3.135 reference picture list 1:** The second *reference picture list* used for *inter prediction* of a *B slice*.

- 3.136 reference picture set (RPS):** A set of *reference pictures* associated with a *picture*, consisting of all *reference pictures* that are prior to the associated *picture* in *decoding order*, that may be used for *inter prediction* of the associated *picture* or any *picture* following the associated *picture* in *decoding order*.
- NOTE – The RPS of a picture consists of five RPS lists, three of which are to contain short-term reference pictures and the other two are to contain long-term reference pictures.
- 3.137 region-wise packed picture:** A decoded picture that contains one or more *packed regions*.
- NOTE – A region-wise packed picture may contain a *region-wise packing* of a *projected picture*.
- 3.138 region-wise packing:** A transformation, resizing, and relocation of *packed regions* of a *region-wise packed picture* to remap the *packed regions* to *projected regions* of a *projected picture*.
- 3.139 reserved:** A term that may be used to specify that some values of a particular *syntax element* are for future use by ITU-T | ISO/IEC and shall not be used in *bitstreams* conforming to this version of this Specification, but may be used in *bitstreams* conforming to future extensions of this Specification by ITU-T | ISO/IEC.
- 3.140 residual:** The decoded difference between a *prediction* of a sample or data element and its decoded value.
- 3.141 sample aspect ratio:** The ratio between the intended horizontal distance between the columns and the intended vertical distance between the rows of the *luma* sample array in a *picture*, which is specified for assisting the *display process* (not specified in this Specification) and expressed as $h:v$, where h is the horizontal width and v is the vertical height, in arbitrary units of spatial distance.
- 3.142 scalable-nested SEI message:** An SEI message that is contained in a scalable nesting SEI message.
- 3.143 scaling:** The process of multiplying *transform coefficient levels* by a factor, resulting in *transform coefficients*.
- 3.144 sequence parameter set (SPS):** A *syntax structure* containing *syntax elements* that apply to zero or more entire *CVSs* as determined by the content of a *syntax element* found in the *PPS* referred to by a *syntax element* found in each *slice segment header*.
- 3.145 shall:** A term used to express mandatory requirements for conformance to this Specification.
- NOTE – When used to express a mandatory constraint on the values of syntax elements or on the results obtained by operation of the specified decoding process, it is the responsibility of the encoder to ensure that the constraint is fulfilled. When used in reference to operations performed by the decoding process, any decoding process that produces identical cropped decoded pictures to those output from the decoding process described in this Specification conforms to the decoding process requirements of this Specification.
- 3.146 short-term reference picture:** A *picture* that is marked as "used for short-term reference".
- 3.147 short-term reference picture set:** The three RPS lists that may contain short-term reference pictures.
- 3.148 should:** A term used to refer to behaviour of an implementation that is encouraged to be followed under anticipated ordinary circumstances, but is not a mandatory requirement for conformance to this Specification.
- 3.149 slice:** An integer number of *CTUs* contained in one *independent slice segment* and all subsequent *dependent slice segments* (if any) that precede the next *independent slice segment* (if any) within the same *access unit*.
- 3.150 slice header:** The *slice segment header* of the *independent slice segment* that is a current *slice segment* or the most recent *independent slice segment* that precedes a current *dependent slice segment* in *decoding order*.
- 3.151 slice segment:** An integer number of *CTUs* ordered consecutively in the *tile scan* and contained in a single *NAL unit*.
- 3.152 slice segment header:** A part of a coded *slice segment* containing the data elements pertaining to the first or all *CTUs* represented in the *slice segment*.
- 3.153 source:** A term used to describe the video material or some of its attributes before encoding.
- 3.154 sphere coordinates:** The azimuth and elevation angles identifying a location of a point on a sphere.
- 3.155 sphere region:** A region on a sphere, specified either by four *great circles* or by two *azimuth circles* and two *elevation circles*, or such a region on a rotated sphere after applying yaw, pitch, and roll rotations.
- 3.156 start code prefix:** A unique sequence of three *bytes* equal to 0x000001 embedded in the *byte stream* as a prefix to each *NAL unit*.
- NOTE – The location of a start code prefix can be used by a decoder to identify the beginning of a new NAL unit and the end of a previous NAL unit. Emulation of start code prefixes is prevented within NAL units by the inclusion of emulation prevention bytes.
- 3.157 step-wise temporal sub-layer access (STSA) access unit:** An *access unit* in which the *coded picture* with *nuh_layer_id* equal to 0 is an *STSA picture*.

- 3.158 step-wise temporal sub-layer access (STSA) picture:** A *coded picture* for which each *VCL NAL unit* has *nal_unit_type* equal to STSA_R or STSA_N.
- NOTE – An STSA picture does not use pictures with the same TemporalId as the STSA picture for inter prediction reference. Pictures following an STSA picture in decoding order with the same TemporalId as the STSA picture do not use pictures prior to the STSA picture in decoding order with the same TemporalId as the STSA picture for inter prediction reference. An STSA picture enables up-switching, at the STSA picture, to the sub-layer containing the STSA picture, from the immediately lower sub-layer. STSA pictures must have TemporalId greater than 0.
- 3.159 string of data bits (SODB):** A sequence of some number of bits representing *syntax elements* present within a *raw byte sequence payload* prior to the *raw byte sequence payload stop bit*, where the left-most bit is considered to be the first and most significant bit, and the right-most bit is considered to be the last and least significant bit.
- 3.160 sub-bitstream extraction process:** A specified process by which *NAL units* in a *bitstream* that do not belong to a target set, determined by a target highest TemporalId and a target *layer identifier list*, are removed from the *bitstream*, with the output sub-bitstream consisting of the *NAL units* in the *bitstream* that belong to the target set.
- 3.161 sub-layer:** A temporal scalable layer of a temporal scalable *bitstream*, consisting of *VCL NAL units* with a particular value of the TemporalId variable and the associated *non-VCL NAL units*.
- 3.162 sub-layer non-reference (SLNR) picture:** A *picture* that contains samples that cannot be used for *inter prediction* in the *decoding process* of subsequent *pictures* of the same *sub-layer* in *decoding order*.
- NOTE – Samples of an SLNR picture may be used for inter prediction in the decoding process of subsequent pictures of higher sub-layers in decoding order.
- 3.163 sub-layer reference picture:** A *picture* that contains samples that may be used for *inter prediction* in the *decoding process* of subsequent *pictures* of the same *sub-layer* in *decoding order*.
- NOTE – Samples of a sub-layer reference picture may also be used for inter prediction in the decoding process of subsequent pictures of higher sub-layers in decoding order.
- 3.164 sub-layer representation:** A subset of the *bitstream* consisting of *NAL units* of a particular *sub-layer* and the lower *sub-layers*.
- 3.165 suffix SEI message:** An SEI message that is contained in a *suffix SEI NAL unit*.
- 3.166 suffix SEI NAL unit:** An *SEI NAL unit* that has *nal_unit_type* equal to SUFFIX_SEI_NUT.
- 3.167 supplemental enhancement information (SEI) NAL unit:** A *NAL unit* that has *nal_unit_type* equal to PREFIX_SEI_NUT or SUFFIX_SEI_NUT.
- 3.168 syntax element:** An element of data represented in the *bitstream*.
- 3.169 syntax structure:** Zero or more *syntax elements* present together in the *bitstream* in a specified order.
- 3.170 temporal sub-layer:** Same as *sub-layer*.
- 3.171 temporal sub-layer access (TSA) access unit:** An *access unit* in which the *coded picture* with *nuh_layer_id* equal to 0 is a *TSA picture*.
- 3.172 temporal sub-layer access (TSA) picture:** A *coded picture* for which each *VCL NAL unit* has *nal_unit_type* equal to TSA_R or TSA_N.
- NOTE – A TSA picture and pictures following the TSA picture in decoding order do not use pictures prior to the TSA picture in decoding order with TemporalId greater than or equal to that of the TSA picture for inter prediction reference. A TSA picture enables up-switching, at the TSA picture, to the sub-layer containing the TSA picture or any higher sub-layer, from the immediately lower sub-layer. TSA pictures must have TemporalId greater than 0.
- 3.173 tier:** A specified category of *level* constraints imposed on values of the *syntax elements* in the *bitstream*, where the *level* constraints are nested within a *tier* and a *decoder* conforming to a certain *tier* and *level* would be capable of decoding all *bitstreams* that conform to the same *tier* or the lower *tier* of that *level* or any *level* below it.
- 3.174 tile:** A rectangular region of *CTUs* within a particular *tile column* and a particular *tile row* in a *picture*.
- 3.175 tile column:** A rectangular region of *CTUs* having a height equal to the height of the *picture* and a width specified by *syntax elements* in the *picture parameter set*.
- 3.176 tile row:** A rectangular region of *CTUs* having a height specified by *syntax elements* in the *picture parameter set* and a width equal to the width of the *picture*.
- 3.177 tile scan:** A specific sequential ordering of *CTUs* partitioning a *picture* in which the *CTUs* are ordered consecutively in *CTU raster scan* in a *tile*, whereas *tiles* in a *picture* are ordered consecutively in a *raster scan* of the *tiles* of the *picture*.

- 3.178 tilt angle:** The angle indicating the amount of tilt of a *sphere region*, measured as the amount of rotation of a *sphere region* along the axis originating from the sphere origin passing through the centre point of the *sphere region*, where the angle value increases clockwise when looking from the origin towards the positive end of the axis.
- 3.179 trailing picture:** A non-IRAP *picture* that follows the *associated IRAP picture* in *output order*.
NOTE – Trailing pictures associated with an IRAP picture also follow the IRAP picture in decoding order. Pictures that follow the associated IRAP picture in output order and precede the associated IRAP picture in decoding order are not allowed.
- 3.180 transform:** A part of the *decoding process* by which a *block* of *transform coefficients* is converted to a *block* of spatial-domain values.
- 3.181 transform block:** A rectangular MxN *block* of samples resulting from a *transform* in the *decoding process*.
- 3.182 transform coefficient:** A scalar quantity, considered to be in a frequency domain, that is associated with a particular one-dimensional or two-dimensional *frequency index* in a *transform* in the *decoding process*.
- 3.183 transform coefficient level:** An integer quantity representing the value associated with a particular two-dimensional frequency index in the *decoding process* prior to *scaling* for computation of a *transform coefficient* value.
- 3.184 transform unit:** A *transform block* of *luma* samples of size 8x8, 16x16, or 32x32 or four *transform blocks* of *luma* samples of size 4x4, two corresponding *transform blocks* of *chroma* samples of a *picture* in 4:2:0 colour format; or a *transform block* of *luma* samples of size 8x8, 16x16, or 32x32, and four corresponding *transform blocks* of *chroma* samples, or four *transform blocks* of *luma* samples of size 4x4, and four corresponding *transform blocks* of *chroma* samples of a *picture* in 4:2:2 colour format; or a *transform block* of *luma* samples of size 4x4, 8x8, 16x16, or 32x32, and two corresponding *transform blocks* of *chroma* samples of a *picture* in 4:4:4 colour format that is not coded using three separate colour planes and *syntax structures* used to transform the *transform block* samples; or a *transform block* of *luma* samples of size 8x8, 16x16, or 32x32 or four *transform blocks* of *luma* samples of size 4x4 of a monochrome *picture* or a *picture* in 4:4:4 colour format that is coded using three separate colour planes; and the associated *syntax structures* used to transform the *transform block* samples.
- 3.185 tree:** A tree is a finite set of nodes with a unique root node.
- 3.186 universal unique identifier (UUID):** An identifier that is unique with respect to the space of all universal unique identifiers.
- 3.187 unspecified:** A term that may be used to specify some values of a particular *syntax element* to indicate that the values have no specified meaning in this Specification and will not have a specified meaning in the future as an integral part of future versions of this Specification.
- 3.188 video coding layer (VCL) NAL unit:** A collective term for *coded slice segment NAL units* and the subset of *NAL units* that have *reserved* values of *nal_unit_type* that are classified as VCL NAL units in this Specification.
- 3.189 video parameter set (VPS):** A *syntax structure* containing *syntax elements* that apply to zero or more entire *CVSs* as determined by the content of a *syntax element* found in the *SPS* referred to by a *syntax element* found in the *PPS* referred to by a *syntax element* found in each *slice segment header*.
- 3.190 viewport:** A region of *omnidirectional video* content suitable for display and viewing by the user.
- 3.191 z-scan order:** A specified sequential ordering of *blocks* partitioning a *picture*, where the order is identical to *CTB raster scan* of the *picture* when the *blocks* are of the same size as *CTBs*, and, when the *blocks* are of a smaller size than *CTBs*, i.e., *CTBs* are further partitioned into smaller *coding blocks*, the order traverses from *CTB* to *CTB* in *CTB raster scan* of the *picture*, and inside each *CTB*, which may be divided into *quadtrees* hierarchically to lower levels, the order traverses from *quadtree* to *quadtree* of a particular level in *quadtree-of-the-particular-level raster scan* of the *quadtree* of the immediately higher level.

4 Abbreviations and acronyms

For the purposes of this Recommendation | International Standard, the following abbreviations and acronyms apply:

ATSC	Advanced Television Systems Committee
B	Bi-predictive
BLA	Broken Link Access

BPB	Bitstream Partition Buffer
CABAC	Context-based Adaptive Binary Arithmetic Coding
CB	Coding Block
CBR	Constant Bit Rate
CIE	International Commission on Illumination (Commission Internationale de l'Eclairage)
CLVS	Coded Layer-wise Video Sequence
CPB	Coded Picture Buffer
CRA	Clean Random Access
CRC	Cyclic Redundancy Check
CTB	Coding Tree Block
CTU	Coding Tree Unit
CU	Coding Unit
CVS	Coded Video Sequence
CVSG	Coded Video Sequence Group
DCT	Discrete Cosine Transform
DPB	Decoded Picture Buffer
DRAP	Dependent Random Access Point
DUT	Decoder Under Test
EG	Exponential-Golomb
EGk	k-th order Exponential-Golomb
FCC	Federal Communications Commission (of the United States)
FIFO	First-In, First-Out
FIR	Finite Impulse Response
FL	Fixed-Length
GBR	Green, Blue and Red
GDR	Gradual Decoding Refresh
HRD	Hypothetical Reference Decoder
HSS	Hypothetical Stream Scheduler
I	Intra
IDCT	Inverse Discrete Cosine Transformation
IDR	Instantaneous Decoding Refresh
INBLD	Independent Non-Base Layer Decoding
IRAP	Intra Random Access Point
LPS	Least Probable Symbol
LSB	Least Significant Bit
MCTS	Motion-Constrained Tile Set
MAC	Multiplexed Analogue Components
MPS	Most Probable Symbol
MSB	Most Significant Bit
MVP	Motion Vector Prediction

NAL	Network Abstraction Layer
NTSC	National Television System Committee (of the United States)
OLS	Output Layer Set
P	Predictive
PAL	Phase Alternating Line
PB	Prediction Block
PCM	Pulse Code Modulation
POC	Picture Order Count
PPS	Picture Parameter Set
PU	Prediction Unit
QP	Quantization Parameter
RADL	Random Access Decodable Leading (Picture)
RASL	Random Access Skipped Leading (Picture)
RBSP	Raw Byte Sequence Payload
RGB	Same as GBR
RPS	Reference Picture Set
SAO	Sample Adaptive Offset
SAR	Sample Aspect Ratio
SECAM	Sequential colour with memory (Séquentiel Couleur avec Mémoire)
SEI	Supplemental Enhancement Information
SLNR	Sub-Layer Non-Reference (Picture)
SMPTE	Society of Motion Picture and Television Engineers
SODB	String Of Data Bits
SPS	Sequence Parameter Set
STSA	Step-wise Temporal Sub-layer Access
TB	Transform Block
TR	Truncated Rice
TSA	Temporal Sub-layer Access
TU	Transform Unit
UCS	Universal Coded Character Set
UTF	UCS Transmission Format
UUID	Universal Unique Identifier
VBR	Variable Bit Rate
VCL	Video Coding Layer
VPS	Video Parameter Set
VUI	Video Usability Information

5 Conventions

5.1 General

NOTE – The mathematical operators used in this Specification are similar to those used in the C programming language. However, the results of integer division and arithmetic shift operations are defined more precisely, and additional operations are defined, such as exponentiation and real-valued division. Numbering and counting conventions generally begin from 0, e.g., "the first" is equivalent to the 0-th, "the second" is equivalent to the 1-th, etc.

5.2 Arithmetic operators

The following arithmetic operators are defined as follows:

+	Addition
–	Subtraction (as a two-argument operator) or negation (as a unary prefix operator)
*	Multiplication, including matrix multiplication
x^y	Exponentiation. Specifies x to the power of y . In other contexts, such notation is used for superscripting not intended for interpretation as exponentiation.
/	Integer division with truncation of the result toward zero. For example, $7 / 4$ and $-7 / -4$ are truncated to 1 and $-7 / 4$ and $7 / -4$ are truncated to -1 .
÷	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\frac{x}{y}$	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\sum_{i=x}^y$	The summation of $f(i)$ with i taking all integer values from x up to and including y .
$x \% y$	Modulus. Remainder of x divided by y , defined only for integers x and y with $x \geq 0$ and $y > 0$.

5.3 Logical operators

The following logical operators are defined as follows:

$x \ \&\& \ y$	Boolean logical "and" of x and y
$x \ \ y$	Boolean logical "or" of x and y
!	Boolean logical "not"
$x ? y : z$	If x is TRUE or not equal to 0, evaluates to the value of y ; otherwise, evaluates to the value of z .

5.4 Relational operators

The following relational operators are defined as follows:

>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
==	Equal to
!=	Not equal to

When a relational operator is applied to a syntax element or variable that has been assigned the value "na" (not applicable), the value "na" is treated as a distinct value for the syntax element or variable. The value "na" is considered not to be equal to any other value.

5.5 Bit-wise operators

The following bit-wise operators are defined as follows:

&	Bit-wise "and". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.
---	---

	Bit-wise "or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.
^	Bit-wise "exclusive or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.
x >> y	Arithmetic right shift of a two's complement integer representation of x by y binary digits. This function is defined only for non-negative integer values of y. Bits shifted into the most significant bits (MSBs) as a result of the right shift have a value equal to the MSB of x prior to the shift operation.
x << y	Arithmetic left shift of a two's complement integer representation of x by y binary digits. This function is defined only for non-negative integer values of y. Bits shifted into the least significant bits (LSBs) as a result of the left shift have a value equal to 0.

5.6 Assignment operators

The following arithmetic operators are defined as follows:

=	Assignment operator
++	Increment, i.e., $x++$ is equivalent to $x = x + 1$; when used in an array index, evaluates to the value of the variable prior to the increment operation.
--	Decrement, i.e., $x--$ is equivalent to $x = x - 1$; when used in an array index, evaluates to the value of the variable prior to the decrement operation.
+=	Increment by amount specified, i.e., $x += 3$ is equivalent to $x = x + 3$, and $x += (-3)$ is equivalent to $x = x + (-3)$.
-=	Decrement by amount specified, i.e., $x -= 3$ is equivalent to $x = x - 3$, and $x -= (-3)$ is equivalent to $x = x - (-3)$.

5.7 Range notation

The following notation is used to specify a range of values:

$x = y..z$ x takes on integer values starting from y to z, inclusive, with x, y, and z being integer numbers and z being greater than or equal to y.

5.8 Mathematical functions

The following mathematical functions are defined:

$$\text{Abs}(x) = \begin{cases} x & ; \quad x \geq 0 \\ -x & ; \quad x < 0 \end{cases} \quad (5-1)$$

$\text{Asin}(x)$ the trigonometric inverse sine function, operating on an argument x that is in the range of -1.0 to 1.0 , inclusive, with an output value in the range of $-\pi/2$ to $\pi/2$, inclusive, in units of radians
(5-2)

$\text{Atan}(x)$ the trigonometric inverse tangent function, operating on an argument x, with an output value in the range of $-\pi/2$ to $\pi/2$, inclusive, in units of radians
(5-3)

$$\text{Atan2}(y, x) = \begin{cases} \text{Atan}\left(\frac{y}{x}\right) & ; \quad x > 0 \\ \text{Atan}\left(\frac{y}{x}\right) + \pi & ; \quad x < 0 \ \&\& \ y \geq 0 \\ \text{Atan}\left(\frac{y}{x}\right) - \pi & ; \quad x < 0 \ \&\& \ y < 0 \\ +\frac{\pi}{2} & ; \quad x == 0 \ \&\& \ y \geq 0 \\ -\frac{\pi}{2} & ; \quad \text{otherwise} \end{cases} \quad (5-4)$$

Ceil(x) the smallest integer greater than or equal to x. (5-5)

Clip1_Y(x) = Clip3(0, (1 << BitDepth_Y) – 1, x)
(5-6)

Clip1_C(x) = Clip3(0, (1 << BitDepth_C) – 1, x)
(5-7)

$$\text{Clip3}(x, y, z) = \begin{cases} x & ; \quad z < x \\ y & ; \quad z > y \\ z & ; \quad \text{otherwise} \end{cases} \quad (5-8)$$

Cos(x) the trigonometric cosine function operating on an argument x in units of radians. (5-9)

Floor(x) the largest integer less than or equal to x.
(5-10)

$$\text{GetCurrMsb}(a, b, c, d) = \begin{cases} c + d & ; \quad b - a \geq d / 2 \\ c - d & ; \quad a - b \geq d / 2 \\ c & ; \quad \text{otherwise} \end{cases} \quad (5-11)$$

Ln(x) the natural logarithm of x (the base-e logarithm, where e is the natural logarithm base constant 2.718 281 828...). (5-12)

Log2(x) the base-2 logarithm of x. (5-13)

Log10(x) the base-10 logarithm of x. (5-14)

$$\text{Min}(x, y) = \begin{cases} x & ; \quad x \leq y \\ y & ; \quad x > y \end{cases} \quad (5-15)$$

$$\text{Max}(x, y) = \begin{cases} x & ; \quad x \geq y \\ y & ; \quad x < y \end{cases} \quad (5-16)$$

Round(x) = Sign(x) * Floor(Abs(x) + 0.5) (5-17)

$$\text{Sign}(x) = \begin{cases} 1 & ; \quad x > 0 \\ 0 & ; \quad x == 0 \\ -1 & ; \quad x < 0 \end{cases} \quad (5-18)$$

Sin(x) the trigonometric sine function operating on an argument x in units of radians (5-19)

Sqrt(x) the square root of x (5-20)

Swap(x, y) = (y, x) (5-21)

Tan(x) the trigonometric tangent function operating on an argument x in units of radians (5-22)

5.9 Order of operation precedence

When order of precedence in an expression is not indicated explicitly by use of parentheses, the following rules apply:

- Operations of a higher precedence are evaluated before any operation of a lower precedence.
- Operations of the same precedence are evaluated sequentially from left to right.

Table 5-1 specifies the precedence of operations from highest to lowest; a higher position in the table indicates a higher precedence.

NOTE – For those operators that are also used in the C programming language, the order of precedence used in this Specification is the same as used in the C programming language.

Table 5-1 – Operation precedence from highest (at top of table) to lowest (at bottom of table)

operations (with operands x, y, and z)
"x++", "x--"
"!x", "-x" (as a unary prefix operator)
x^y
"x * y", "x / y", "x ÷ y", " $\frac{x}{y}$ ", "x % y"
"x + y", "x - y" (as a two-argument operator), " $\sum_{i=x}^y f(i)$ "
"x << y", "x >> y"
"x < y", "x <= y", "x > y", "x >= y"
"x == y", "x != y"
"x & y"
"x y"
"x && y"
"x y"
"x ? y : z"
"x.y"
"x = y", "x += y", "x -= y"

5.10 Variables, syntax elements and tables

Syntax elements in the bitstream are represented in **bold** type. Each syntax element is described by its name (all lower case letters with underscore characters), and one descriptor for its method of coded representation. The decoding process behaves according to the value of the syntax element and to the values of previously decoded syntax elements. When a value of a syntax element is used in the syntax tables or the text, it appears in regular (i.e., not bold) type.

In some cases the syntax tables may use the values of other variables derived from syntax elements values. Such variables appear in the syntax tables, or text, named by a mixture of lower case and upper case letters and without any underscore characters. Variables starting with an upper case letter are derived for the decoding of the current syntax structure and all depending syntax structures. Variables starting with an upper case letter may be used in the decoding process for later syntax structures without mentioning the originating syntax structure of the variable. Variables starting with a lower case letter are only used within the clause in which they are derived.

In some cases, "mnemonic" names for syntax element values or variable values are used interchangeably with their numerical values. Sometimes "mnemonic" names are used without any associated numerical values. The association of values and names is specified in the text. The names are constructed from one or more groups of letters separated by an underscore character. Each group starts with an upper case letter and may contain more upper case letters.

NOTE – The syntax is described in a manner that closely follows the C-language syntactic constructs.

Functions that specify properties of the current position in the bitstream are referred to as syntax functions. These functions are specified in clause 7.2 and assume the existence of a bitstream pointer with an indication of the position of the next bit to be read by the decoding process from the bitstream. Syntax functions are described by their names, which are constructed as syntax element names and end with left and right parentheses including zero or more variable names (for definition) or values (for usage), separated by commas (if more than one variable).

Functions that are not syntax functions (including mathematical functions specified in clause 5.8) are described by their names, which start with an upper case letter, contain a mixture of lower and upper case letters without any underscore character, and end with left and right parentheses including zero or more variable names (for definition) or values (for usage) separated by commas (if more than one variable).

A one-dimensional array is referred to as a list. A two-dimensional array is referred to as a matrix. Arrays can either be syntax elements or variables. Subscripts or square parentheses are used for the indexing of arrays. In reference to a visual depiction of a matrix, the first subscript is used as a row (vertical) index and the second subscript is used as a column (horizontal) index. The indexing order is reversed when using square parentheses rather than subscripts for indexing.

Thus, an element of a matrix s at horizontal position x and vertical position y may be denoted either as $s[x][y]$ or as s_{yx} . A single column of a matrix may be referred to as a list and denoted by omission of the row index. Thus, the column of a matrix s at horizontal position x may be referred to as the list $s[x]$.

A specification of values of the entries in rows and columns of an array may be denoted by $\{ \{ \dots \} \{ \dots \} \}$, where each inner pair of brackets specifies the values of the elements within a row in increasing column order and the rows are ordered in increasing row order. Thus, setting a matrix s equal to $\{ \{ 1 \ 6 \} \{ 4 \ 9 \} \}$ specifies that $s[0][0]$ is set equal to 1, $s[1][0]$ is set equal to 6, $s[0][1]$ is set equal to 4, and $s[1][1]$ is set equal to 9.

Binary notation is indicated by enclosing the string of bit values by single quote marks. For example, '01000001' represents an eight-bit string having only its second and its last bits (counted from the most to the least significant bit) equal to 1.

Hexadecimal notation, indicated by prefixing the hexadecimal number by "0x", may be used instead of binary notation when the number of bits is an integer multiple of 4. For example, 0x41 represents an eight-bit string having only its second and its last bits (counted from the most to the least significant bit) equal to 1.

Numerical values not enclosed in single quotes and not prefixed by "0x" are decimal values.

A value equal to 0 represents a FALSE condition in a test statement. The value TRUE is represented by any value different from zero.

5.11 Text description of logical operations

In the text, a statement of logical operations as would be described mathematically in the following form:

```
if( condition 0 )
    statement 0
else if( condition 1 )
    statement 1
...
else /* informative remark on remaining condition */
    statement n
```

may be described in the following manner:

- ... as follows / ... the following applies:
- If condition 0, statement 0
- Otherwise, if condition 1, statement 1
- ...
- Otherwise (informative remark on remaining condition), statement n

Each "If ... Otherwise, if ... Otherwise, ..." statement in the text is introduced with "... as follows" or "... the following applies" immediately followed by "If ... ". The last condition of the "If ... Otherwise, if ... Otherwise, ..." is always an "Otherwise, ...". Interleaved "If ... Otherwise, if ... Otherwise, ..." statements can be identified by matching "... as follows" or "... the following applies" with the ending "Otherwise, ...".

In the text, a statement of logical operations as would be described mathematically in the following form:

```
if( condition 0a && condition 0b )
    statement 0
else if( condition 1a || condition 1b )
    statement 1
...
else
    statement n
```

may be described in the following manner:

- ... as follows / ... the following applies:
- If all of the following conditions are true, statement 0:
 - condition 0a
 - condition 0b
- Otherwise, if one or more of the following conditions are true, statement 1:

- condition 1a
- condition 1b
- ...
- Otherwise, statement n

In the text, a statement of logical operations as would be described mathematically in the following form:

```

if( condition 0 )
    statement 0
if( condition 1 )
    statement 1

```

may be described in the following manner:

```

When condition 0, statement 0
When condition 1, statement 1

```

5.12 Processes

Processes are used to describe the decoding of syntax elements. A process has a separate specification and invoking. All syntax elements and upper case variables that pertain to the current syntax structure and depending syntax structures are available in the process specification and invoking. A process specification may also have a lower case variable explicitly specified as input. Each process specification has explicitly specified an output. The output is a variable that can either be an upper case variable or a lower case variable.

When invoking a process, the assignment of variables is specified as follows:

- If the variables at the invoking and the process specification do not have the same name, the variables are explicitly assigned to lower case input or output variables of the process specification.
- Otherwise (the variables at the invoking and the process specification have the same name), assignment is implied.

In the specification of a process, a specific coding block may be referred to by the variable name having a value equal to the address of the specific coding block.

6 Bitstream and picture formats, partitionings, scanning processes and neighbouring relationships

6.1 Bitstream formats

This clause specifies the relationship between the network abstraction layer (NAL) unit stream and byte stream, either of which are referred to as the bitstream.

The bitstream can be in one of two formats: the NAL unit stream format or the byte stream format. The NAL unit stream format is conceptually the more "basic" type. It consists of a sequence of syntax structures called NAL units. This sequence is ordered in decoding order. There are constraints imposed on the decoding order (and contents) of the NAL units in the NAL unit stream.

The byte stream format can be constructed from the NAL unit stream format by ordering the NAL units in decoding order and prefixing each NAL unit with a start code prefix and zero or more zero-valued bytes to form a stream of bytes. The NAL unit stream format can be extracted from the byte stream format by searching for the location of the unique start code prefix pattern within this stream of bytes. Methods of framing the NAL units in a manner other than use of the byte stream format are outside the scope of this Specification. The byte stream format is specified in Annex B.

6.2 Source, decoded and output picture formats

This clause specifies the relationship between source and decoded pictures that is given via the bitstream.

The video source that is represented by the bitstream is a sequence of pictures in decoding order.

The source and decoded pictures are each comprised of one or more sample arrays:

- Luma (Y) only (monochrome).
- Luma and two chroma (YCbCr or YCgCo).

- Green, blue, and red (GBR, also known as RGB).
- Arrays representing other unspecified monochrome or tri-stimulus colour samplings (for example, YZX, also known as XYZ).

For convenience of notation and terminology in this Specification, the variables and terms associated with these arrays are referred to as luma (or L or Y) and chroma, where the two chroma arrays are referred to as Cb and Cr; regardless of the actual colour representation method in use. The actual colour representation method in use can be indicated in syntax that is specified in Annex E.

The variables SubWidthC and SubHeightC are specified in Table 6-1, depending on the chroma format sampling structure, which is specified through chroma_format_idc and separate_colour_plane_flag. Other values of chroma_format_idc, SubWidthC and SubHeightC may be specified in the future by ITU-T | ISO/IEC.

Table 6-1 – SubWidthC and SubHeightC values derived from chroma_format_idc and separate_colour_plane_flag

chroma_format_idc	separate_colour_plane_flag	Chroma format	SubWidthC	SubHeightC
0	0	Monochrome	1	1
1	0	4:2:0	2	2
2	0	4:2:2	2	1
3	0	4:4:4	1	1
3	1	4:4:4	1	1

In monochrome sampling there is only one sample array, which is nominally considered the luma array.

In 4:2:0 sampling, each of the two chroma arrays has half the height and half the width of the luma array.

In 4:2:2 sampling, each of the two chroma arrays has the same height and half the width of the luma array.

In 4:4:4 sampling, depending on the value of separate_colour_plane_flag, the following applies:

- If separate_colour_plane_flag is equal to 0, each of the two chroma arrays has the same height and width as the luma array.
- Otherwise (separate_colour_plane_flag is equal to 1), the three colour planes are separately processed as monochrome sampled pictures.

The number of bits necessary for the representation of each of the samples in the luma and chroma arrays in a video sequence is in the range of 8 to 16, inclusive, and the number of bits used in the luma array may differ from the number of bits used in the chroma arrays.

When the value of chroma_format_idc is equal to 1, the nominal vertical and horizontal relative locations of luma and chroma samples in pictures are shown in Figure 6-1. Alternative chroma sample relative locations may be indicated in video usability information (see Annex E).

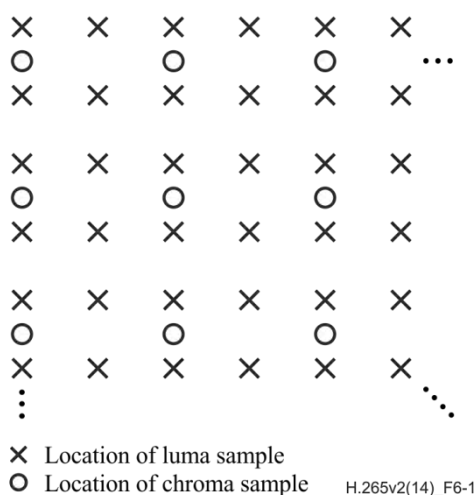


Figure 6-1 – Nominal vertical and horizontal locations of 4:2:0 luma and chroma samples in a picture

When the value of chroma_format_idc is equal to 2, the chroma samples are co-sited with the corresponding luma samples and the nominal locations in a picture are as shown in Figure 6-2.

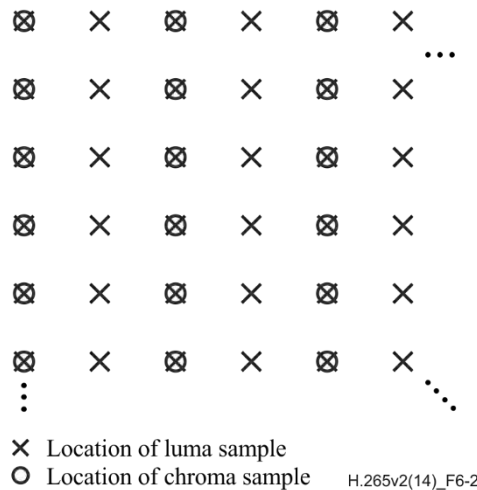


Figure 6-2 – Nominal vertical and horizontal locations of 4:2:2 luma and chroma samples in a picture

When the value of chroma_format_idc is equal to 3, all array samples are co-sited for all cases of pictures and the nominal locations in a picture are as shown in Figure 6-3.

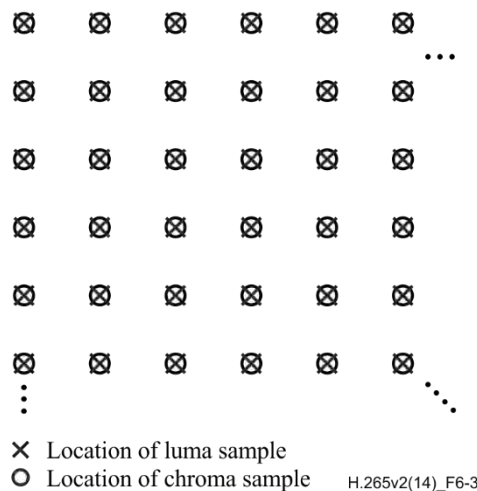


Figure 6-3 – Nominal vertical and horizontal locations of 4:4:4 luma and chroma samples in a picture

6.3 Partitioning of pictures, slices, slice segments, tiles, CTUs and CTBs

6.3.1 Partitioning of pictures into slices, slice segments and tiles

This clause specifies how a picture is partitioned into slices, slice segments and tiles. Pictures are divided into slices and tiles. A slice is a sequence of one or more slice segments starting with an independent slice segment and containing all subsequent dependent slice segments (if any) that precede the next independent slice segment (if any) within the same picture. A slice segment is a sequence of CTUs. Likewise, a tile is a sequence of CTUs.

For example, a picture may be divided into two slices as shown in Figure 6-4. In this example, the first slice is composed of an independent slice segment containing 4 CTUs, a dependent slice segment containing 32 CTUs and another dependent slice segment containing 24 CTUs; and the second slice consists of a single independent slice segment containing the remaining 39 CTUs of the picture.

As another example, a picture may be divided into two tiles separated by a vertical tile boundary as shown in Figure 6-5. The left side of the figure illustrates a case in which the picture only contains one slice, starting with an independent slice segment and followed by four dependent slice segments. The right side of the figure illustrates an alternative case in which the picture contains two slices in the first tile and one slice in the second tile.

Unlike slices, tiles are always rectangular. A tile always contains an integer number of CTUs, and may consist of CTUs contained in more than one slice. Similarly, a slice may consist of CTUs contained in more than one tile.

One or both of the following conditions shall be fulfilled for each slice and tile:

- All CTUs in a slice belong to the same tile.

- All CTUs in a tile belong to the same slice.

NOTE 1 – Within the same picture, there may be both slices that contain multiple tiles and tiles that contain multiple slices.

One or both of the following conditions shall be fulfilled for each slice segment and tile:

- All CTUs in a slice segment belong to the same tile.
- All CTUs in a tile belong to the same slice segment.

When a picture is coded using three separate colour planes (separate_colour_plane_flag is equal to 1), a slice contains only CTBs of one colour component being identified by the corresponding value of colour_plane_id, and each colour component array of a picture consists of slices having the same colour_plane_id value. Coded slices with different values of colour_plane_id within a picture may be interleaved with each other under the constraint that for each value of colour_plane_id, the coded slice segment NAL units with that value of colour_plane_id shall be in the order of increasing CTB address in tile scan order for the first CTB of each coded slice segment NAL unit.

NOTE 2 – When separate_colour_plane_flag is equal to 0, each CTB of a picture is contained in exactly one slice. When separate_colour_plane_flag is equal to 1, each CTB of a colour component is contained in exactly one slice (i.e., information for each CTB of a picture is present in exactly three slices and these three slices have different values of colour_plane_id).

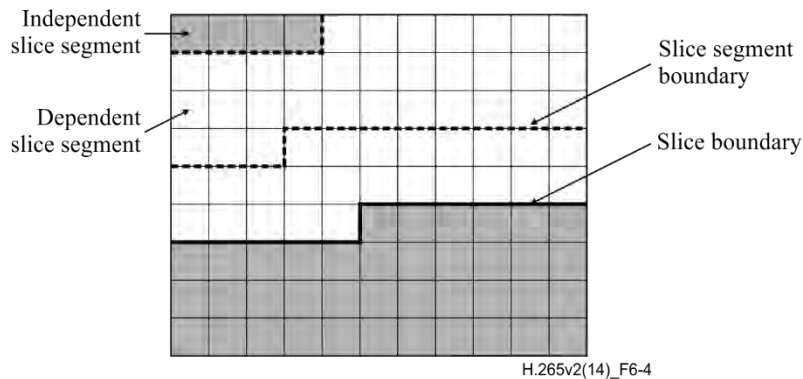


Figure 6-4 – A picture with 11 by 9 luma CTBs that is partitioned into two slices, the first of which is partitioned into three slice segments (informative)

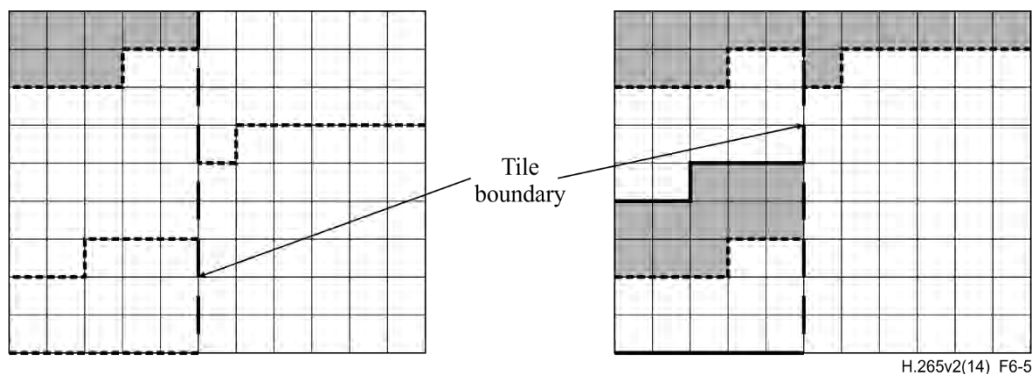


Figure 6-5 – A picture with 11 by 9 luma CTBs that is partitioned into two tiles and one slice (left) or is partitioned into two tiles and three slices (right) (informative)

6.3.2 Block and quadtree structures

The samples are processed in units of CTBs. The array size for each luma CTB in both width and height is CtbSizeY in units of samples. The width and height of the array for each chroma CTB are CtbWidthC and CtbHeightC, respectively, in units of samples.

Each CTB is assigned a partition signalling to identify the block sizes for intra or inter prediction and for transform coding. The partitioning is a recursive quadtree partitioning. The root of the quadtree is associated with the CTB. The quadtree is split until a leaf is reached, which is referred to as the coding block. When the component width is not an integer number of the CTB size, the CTBs at the right component boundary are incomplete. When the component height is not an integer multiple of the CTB size, the CTBs at the bottom component boundary are incomplete.

The coding block is the root node of two trees, the prediction tree and the transform tree. The prediction tree specifies the position and size of prediction blocks. The transform tree specifies the position and size of transform blocks. The splitting information for luma and chroma is identical for the prediction tree and may or may not be identical for the transform tree.

The blocks and associated syntax structures are grouped into "unit" structures as follows:

- One prediction block (monochrome picture or `separate_colour_plane_flag` is equal to 1) or three prediction blocks (luma and chroma components of a picture in 4:2:0 or 4:4:4 colour format) or five prediction blocks (luma and chroma components of a picture in 4:2:2 colour format) and the associated prediction syntax structures units are associated with a prediction unit.
- One transform block (monochrome picture or `separate_colour_plane_flag` is equal to 1) or three transform blocks (luma and chroma components of a picture in 4:2:0 or 4:4:4 colour format) or five transform blocks (luma and chroma components of a picture in 4:2:2 colour format) and the associated transform syntax structures units are associated with a transform unit.
- One coding block (monochrome picture or `separate_colour_plane_flag` is equal to 1) or three coding blocks (luma and chroma), the associated coding syntax structures and the associated prediction and transform units are associated with a coding unit.
- One CTB (monochrome picture or `separate_colour_plane_flag` is equal to 1) or three CTBs (luma and chroma), the associated coding tree syntax structures and the associated coding units are associated with a CTU.

6.3.3 Spatial or component-wise partitionings

The following divisions of processing elements of this Specification form spatial or component-wise partitionings:

- The division of each picture into components
- The division of each component into CTBs
- The division of each picture into tile columns
- The division of each picture into tile rows
- The division of each tile column into tiles
- The division of each tile row into tiles
- The division of each tile into CTUs
- The division of each picture into slices
- The division of each slice into slice segments
- The division of each slice segment into CTUs
- The division of each CTU into CTBs
- The division of each CTB into coding blocks, except that the CTBs are incomplete at the right component boundary when the component width is not an integer multiple of the CTB size and the CTBs are incomplete at the bottom component boundary when the component height is not an integer multiple of the CTB size
- The division of each CTU into coding units, except that the CTUs are incomplete at the right picture boundary when the picture width in luma samples is not an integer multiple of the luma CTB size and the CTUs are incomplete at the bottom picture boundary when the picture height in luma samples is not an integer multiple of the luma CTB size
- The division of each coding unit into prediction units
- The division of each coding unit into transform units
- The division of each coding unit into coding blocks
- The division of each coding block into prediction blocks
- The division of each coding block into transform blocks
- The division of each prediction unit into prediction blocks
- The division of each transform unit into transform blocks.

6.4 Availability processes

6.4.1 Derivation process for z-scan order block availability

Inputs to this process are:

- The luma location (x_{Curr} , y_{Curr}) of the top-left sample of the current block relative to the top-left luma sample of the current picture
- The luma location (x_{NbY} , y_{NbY}) covered by a neighbouring block relative to the top-left luma sample of the current picture.

Output of this process is the availability of the neighbouring block covering the location (x_{NbY} , y_{NbY}), denoted as $availableN$.

The minimum luma block address in z-scan order $minBlockAddrCurr$ of the current block is derived as follows:

$$minBlockAddrCurr = MinTbAddrZs[x_{Curr} \gg MinTbLog2SizeY][y_{Curr} \gg MinTbLog2SizeY] \quad (6-1)$$

The minimum luma block address in z-scan order $minBlockAddrN$ of the neighbouring block covering the location (x_{NbY} , y_{NbY}) is derived as follows:

- If one or more of the following conditions are true, $minBlockAddrN$ is set equal to -1 :
 - x_{NbY} is less than 0
 - y_{NbY} is less than 0
 - x_{NbY} is greater than or equal to $pic_width_in_luma_samples$
 - y_{NbY} is greater than or equal to $pic_height_in_luma_samples$
- Otherwise (x_{NbY} and y_{NbY} are inside the picture boundaries),

$$minBlockAddrN = MinTbAddrZs[x_{NbY} \gg MinTbLog2SizeY][y_{NbY} \gg MinTbLog2SizeY] \quad (6-2)$$

The neighbouring block availability $availableN$ is derived as follows:

- If one or more of the following conditions are true, $availableN$ is set equal to FALSE:
 - $minBlockAddrN$ is less than 0,
 - $minBlockAddrN$ is greater than $minBlockAddrCurr$,
 - the variable $SliceAddrRs$ associated with the slice segment containing the neighbouring block with the minimum luma block address $minBlockAddrN$ differs in value from the variable $SliceAddrRs$ associated with the slice segment containing the current block with the minimum luma block address $minBlockAddrCurr$.
 - the neighbouring block with the minimum luma block address $minBlockAddrN$ is contained in a different tile than the current block with the minimum luma block address $minBlockAddrCurr$.
- Otherwise, $availableN$ is set equal to TRUE.

6.4.2 Derivation process for prediction block availability

Inputs to this process are:

- the luma location (x_{Cb} , y_{Cb}) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a variable $nCbS$ specifying the size of the current luma coding block,
- the luma location (x_{Pb} , y_{Pb}) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
- two variables $nPbW$ and $nPbH$ specifying the width and the height of the current luma prediction block,
- a variable $partIdx$ specifying the partition index of the current prediction unit within the current coding unit,
- the luma location (x_{NbY} , y_{NbY}) covered by a neighbouring prediction block relative to the top-left luma sample of the current picture.

Output of this process is the availability of the neighbouring prediction block covering the location (x_{NbY} , y_{NbY}), denoted as $availableN$ is derived as follows:

The variable $sameCb$ identifies whether the current luma prediction block and the neighbouring luma prediction block cover the same luma coding block, and is derived as follows:

- If all of the following conditions are true, $sameCb$ is set equal to TRUE:
 - x_{Cb} is less than or equal than x_{NbY} ,
 - y_{Cb} is less than or equal than y_{NbY} ,
 - $(x_{Cb} + n_{CbS})$ is greater than x_{NbY} ,
 - $(y_{Cb} + n_{CbS})$ is greater than y_{NbY} .
- Otherwise, $sameCb$ is set equal to FALSE.

The neighbouring prediction block availability $availableN$ is derived as follows:

- If $sameCb$ is equal to FALSE, the derivation process for z-scan order block availability as specified in clause 6.4.1 is invoked with (x_{Curr} , y_{Curr}) set equal to (x_{Pb} , y_{Pb}) and the luma location (x_{NbY} , y_{NbY}) as inputs, and the output is assigned to $availableN$.
- Otherwise, if all of the following conditions are true, $availableN$ is set equal to FALSE:
 - $(n_{PbW} << 1)$ is equal to n_{CbS} ,
 - $(n_{PbH} << 1)$ is equal to n_{CbS} ,
 - $partIdx$ is equal to 1,
 - $(y_{Cb} + n_{PbH})$ is less than or equal to y_{NbY} ,
 - $(x_{Cb} + n_{PbW})$ is greater than x_{NbY} .
- Otherwise, $availableN$ is set equal to TRUE.

When $availableN$ is equal to TRUE and $CuPredMode[x_{NbY}][y_{NbY}]$ is equal to $MODE_INTRA$, $availableN$ is set equal to FALSE.

6.5 Scanning processes

6.5.1 CTB raster and tile scanning conversion process

The list $colWidth[i]$ for i ranging from 0 to $num_tile_columns_minus1$, inclusive, specifying the width of the i -th tile column in units of CTBs, is derived as follows:

```

if( uniform_spacing_flag )
    for( i = 0; i <= num_tile_columns_minus1; i++ )
        colWidth[ i ] = ( ( i + 1 ) * PicWidthInCtbsY ) / ( num_tile_columns_minus1 + 1 ) -
                        ( i * PicWidthInCtbsY ) / ( num_tile_columns_minus1 + 1 )
else {
    colWidth[ num_tile_columns_minus1 ] = PicWidthInCtbsY
    for( i = 0; i < num_tile_columns_minus1; i++ ) {
        colWidth[ i ] = column_width_minus1[ i ] + 1
        colWidth[ num_tile_columns_minus1 ] -= colWidth[ i ]
    }
}

```

(6-3)

The list $rowHeight[j]$ for j ranging from 0 to $num_tile_rows_minus1$, inclusive, specifying the height of the j -th tile row in units of CTBs, is derived as follows:

```

if( uniform_spacing_flag )
    for( j = 0; j <= num_tile_rows_minus1; j++ )
        rowHeight[ j ] = ( ( j + 1 ) * PicHeightInCtbsY ) / ( num_tile_rows_minus1 + 1 ) -
                        ( j * PicHeightInCtbsY ) / ( num_tile_rows_minus1 + 1 )
else {
    rowHeight[ num_tile_rows_minus1 ] = PicHeightInCtbsY

```

(6-4)

```

    for( j = 0; j < num_tile_rows_minus1; j++ ) {
        rowHeight[ j ] = row_height_minus1[ j ] + 1
        rowHeight[ num_tile_rows_minus1 ] -= rowHeight[ j ]
    }
}

```

The list colBd[i] for i ranging from 0 to num_tile_columns_minus1 + 1, inclusive, specifying the location of the i-th tile column boundary in units of CTBs, is derived as follows:

```

for( colBd[ 0 ] = 0, i = 0; i <= num_tile_columns_minus1; i++ )
    colBd[ i + 1 ] = colBd[ i ] + colWidth[ i ]
(6-5)

```

The list rowBd[j] for j ranging from 0 to num_tile_rows_minus1 + 1, inclusive, specifying the location of the j-th tile row boundary in units of CTBs, is derived as follows:

```

for( rowBd[ 0 ] = 0, j = 0; j <= num_tile_rows_minus1; j++ )
    rowBd[ j + 1 ] = rowBd[ j ] + rowHeight[ j ]
(6-6)

```

The list CtbAddrRsToTs[ctbAddrRs] for ctbAddrRs ranging from 0 to PicSizeInCtbsY - 1, inclusive, specifying the conversion from a CTB address in CTB raster scan of a picture to a CTB address in tile scan, is derived as follows:

```

for( ctbAddrRs = 0; ctbAddrRs < PicSizeInCtbsY; ctbAddrRs++ ) {
    tbX = ctbAddrRs % PicWidthInCtbsY
    tbY = ctbAddrRs / PicWidthInCtbsY
    for( i = 0; i <= num_tile_columns_minus1; i++ )
        if( tbX >= colBd[ i ] )
            tileX = i
    for( j = 0; j <= num_tile_rows_minus1; j++ )
        if( tbY >= rowBd[ j ] )
            tileY = j
    CtbAddrRsToTs[ ctbAddrRs ] = 0
    for( i = 0; i < tileX; i++ )
        CtbAddrRsToTs[ ctbAddrRs ] += rowHeight[ tileY ] * colWidth[ i ]
    for( j = 0; j < tileY; j++ )
        CtbAddrRsToTs[ ctbAddrRs ] += PicWidthInCtbsY * rowHeight[ j ]
    CtbAddrRsToTs[ ctbAddrRs ] += ( tbY - rowBd[ tileY ]
) * colWidth[ tileX ] + tbX - colBd[ tileX ]
}
(6-7)

```

The list CtbAddrTsToRs[ctbAddrTs] for ctbAddrTs ranging from 0 to PicSizeInCtbsY - 1, inclusive, specifying the conversion from a CTB address in tile scan to a CTB address in CTB raster scan of a picture, is derived as follows:

```

for( ctbAddrRs = 0; ctbAddrRs < PicSizeInCtbsY; ctbAddrRs++ )
    CtbAddrTsToRs[ CtbAddrRsToTs[ ctbAddrRs ] ] = ctbAddrRs
(6-8)

```

The list TileId[ctbAddrTs] for ctbAddrTs ranging from 0 to PicSizeInCtbsY - 1, inclusive, specifying the conversion from a CTB address in tile scan to a tile ID, is derived as follows:

```

for( j = 0, tileIdx = 0; j <= num_tile_rows_minus1; j++ )
    for( i = 0; i <= num_tile_columns_minus1; i++, tileIdx++ )
        for( y = rowBd[ j ]; y < rowBd[ j + 1 ]; y++ )
            for( x = colBd[ i ]; x < colBd[ i + 1 ]; x++ )
                TileId[ CtbAddrRsToTs[ y * PicWidthInCtbsY + x ] ] = tileIdx
(6-9)

```

The values of ColumnWidthInLumaSamples[i], specifying the width of the i-th tile column in units of luma samples, are set equal to colWidth[i] << CtbLog2SizeY for i ranging from 0 to num_tile_columns_minus1, inclusive.

The values of `RowHeightInLumaSamples[j]`, specifying the height of the j -th tile row in units of luma samples, are set equal to `rowHeight[j] << CtbLog2SizeY` for j ranging from 0 to `num_tile_rows_minus1`, inclusive.

6.5.2 Z-scan order array initialization process

The array `MinTbAddrZs` with elements `MinTbAddrZs[x][y]` for x ranging from 0 to $(\text{PicWidthInCtbsY} \ll (\text{CtbLog2SizeY} - \text{MinTbLog2SizeY})) - 1$, inclusive, and y ranging from 0 to $(\text{PicHeightInCtbsY} \ll (\text{CtbLog2SizeY} - \text{MinTbLog2SizeY})) - 1$, specifying the conversion from a location (x, y) in units of minimum blocks to a minimum block address in z-scan order, inclusive, is derived as follows:

```

for( y = 0; y < ( PicHeightInCtbsY << ( CtbLog2SizeY - MinTbLog2SizeY ) ); y++ )
  for( x = 0; x < ( PicWidthInCtbsY << ( CtbLog2SizeY - MinTbLog2SizeY ) ); x++ ) {
    tbX = ( x << MinTbLog2SizeY ) >> CtbLog2SizeY
    tbY = ( y << MinTbLog2SizeY ) >> CtbLog2SizeY
    ctbAddrRs = PicWidthInCtbsY * tbY + tbX
    MinTbAddrZs[ x ][ y ] = CtbAddrRsToTs[ ctbAddrRs ] <<
      ( ( CtbLog2SizeY - MinTbLog2SizeY ) * 2 )
    for( i = 0, p = 0; i < ( CtbLog2SizeY - MinTbLog2SizeY ); i++ ) {
      m = 1 << i
      p += ( m & x ? m * m : 0 ) + ( m & y ? 2 * m * m : 0 )
    }
    MinTbAddrZs[ x ][ y ] += p
  }

```

(6-10)

6.5.3 Up-right diagonal scan order array initialization process

Input to this process is a block size `blkSize`.

Output of this process is the array `diagScan[sPos][sComp]`. The array index `sPos` specify the scan position ranging from 0 to $(\text{blkSize} * \text{blkSize}) - 1$. The array index `sComp` equal to 0 specifies the horizontal component and the array index `sComp` equal to 1 specifies the vertical component. Depending on the value of `blkSize`, the array `diagScan` is derived as follows:

```

i = 0
x = 0
y = 0
stopLoop = FALSE
while( !stopLoop ) {
  while( y >= 0 ) {
    if( x < blkSize && y < blkSize ) {
      diagScan[ i ][ 0 ] = x
      diagScan[ i ][ 1 ] = y
      i++
    }
    y--
    x++
  }
  y = x
  x = 0
  if( i >= blkSize * blkSize )
    stopLoop = TRUE
}

```

(6-11)

6.5.4 Horizontal scan order array initialization process

Input to this process is a block size `blkSize`.

Output of this process is the array `horScan[sPos][sComp]`. The array index `sPos` specifies the scan position ranging from 0 to $(\text{blkSize} * \text{blkSize}) - 1$. The array index `sComp` equal to 0 specifies the horizontal component and the array index `sComp` equal to 1 specifies the vertical component. Depending on the value of `blkSize`, the array `horScan` is derived as follows:

```

i = 0
for( y = 0; y < blkSize; y++ )
    for( x = 0; x < blkSize; x++ ) {
        horScan[ i ][ 0 ] = x
        horScan[ i ][ 1 ] = y
        i++
    }

```

(6-12)

6.5.5 Vertical scan order array initialization process

Input to this process is a block size blkSize.

Output of this process is the array verScan[sPos][sComp]. The array index sPos specifies the scan position ranging from 0 to (blkSize * blkSize) – 1. The array index sComp equal to 0 specifies the horizontal component and the array index sComp equal to 1 specifies the vertical component. Depending on the value of blkSize, the array verScan is derived as follows:

```

i = 0
for( x = 0; x < blkSize; x++ )
    for( y = 0; y < blkSize; y++ ) {
        verScan[ i ][ 0 ] = x
        verScan[ i ][ 1 ] = y
        i++
    }

```

(6-13)

6.5.6 Traverse scan order array initialization process

Input to this process is a block size blkSize.

Output of this process is the array travScan[sPos][sComp]. The array index sPos specifies the scan position ranging from 0 to (blkSize * blkSize) – 1, inclusive. The array index sComp equal to 0 specifies the horizontal component and the array index sComp equal to 1 specifies the vertical component. Depending on the value of blkSize, the array travScan is derived as follows:

```

i = 0
for( y = 0; y < blkSize; y++ )
    if( y % 2 == 0 )
        for( x = 0; x < blkSize; x++ ) {
            travScan[ i ][ 0 ] = x
            travScan[ i ][ 1 ] = y
            i++
        }
    else
        for( x = blkSize – 1; x >= 0; x-- ) {
            travScan[ i ][ 0 ] = x
            travScan[ i ][ 1 ] = y
            i++
        }

```

(6-14)

7 Syntax and semantics

7.1 Method of specifying syntax in tabular form

The syntax tables specify a superset of the syntax of all allowed bitstreams. Additional constraints on the syntax may be specified, either directly or indirectly, in other clauses.

NOTE – An actual decoder should implement some means for identifying entry points into the bitstream and some means to identify and handle non-conforming bitstreams. The methods for identifying and handling errors and other such situations are not specified in this Specification.

The following table lists examples of the syntax specification format. When **syntax_element** appears, it specifies that a syntax element is parsed from the bitstream and the bitstream pointer is advanced to the next position beyond the syntax element in the bitstream parsing process.

	Descriptor
/* A statement can be a syntax element with an associated descriptor or can be an expression used to specify conditions for the existence, type and quantity of syntax elements, as in the following two examples */	
syntax_element	ue(v)
conditioning statement	
/* A group of statements enclosed in curly brackets is a compound statement and is treated functionally as a single statement. */	
{	
statement	
statement	
...	
}	
/* A "while" structure specifies a test of whether a condition is true, and if true, specifies evaluation of a statement (or compound statement) repeatedly until the condition is no longer true */	
while(condition)	
statement	
/* A "do ... while" structure specifies evaluation of a statement once, followed by a test of whether a condition is true, and if true, specifies repeated evaluation of the statement until the condition is no longer true */	
do	
statement	
while(condition)	
/* An "if ... else" structure specifies a test of whether a condition is true and, if the condition is true, specifies evaluation of a primary statement, otherwise, specifies evaluation of an alternative statement. The "else" part of the structure and the associated alternative statement is omitted if no alternative statement evaluation is needed */	
if(condition)	
primary statement	
else	
alternative statement	
/* A "for" structure specifies evaluation of an initial statement, followed by a test of a condition, and if the condition is true, specifies repeated evaluation of a primary statement followed by a subsequent statement until the condition is no longer true. */	
for(initial statement; condition; subsequent statement)	
primary statement	

7.2 Specification of syntax functions and descriptors

The functions presented here are used in the syntactical description. These functions are expressed in terms of the value of a bitstream pointer that indicates the position of the next bit to be read by the decoding process from the bitstream.

byte_aligned() is specified as follows:

- If the current position in the bitstream is on a byte boundary, i.e., the next bit in the bitstream is the first bit in a byte, the return value of byte_aligned() is equal to TRUE.
- Otherwise, the return value of byte_aligned() is equal to FALSE.

`more_data_in_byte_stream()`, which is used only in the byte stream NAL unit syntax structure specified in Annex B, is specified as follows:

- If more data follow in the byte stream, the return value of `more_data_in_byte_stream()` is equal to TRUE.
- Otherwise, the return value of `more_data_in_byte_stream()` is equal to FALSE.

`more_data_in_payload()` is specified as follows:

- If `byte_aligned()` is equal to TRUE and the current position in the `sei_payload()` syntax structure is $8 * \text{payloadSize}$ bits from the beginning of the `sei_payload()` syntax structure, the return value of `more_data_in_payload()` is equal to FALSE.
- Otherwise, the return value of `more_data_in_payload()` is equal to TRUE.

`more_rbsp_data()` is specified as follows:

- If there is no more data in the raw byte sequence payload (Rbsp), the return value of `more_rbsp_data()` is equal to FALSE.
- Otherwise, the Rbsp data are searched for the last (least significant, right-most) bit equal to 1 that is present in the Rbsp. Given the position of this bit, which is the first bit (`rbp_stop_one_bit`) of the `rbp_trailing_bits()` syntax structure, the following applies:
 - If there is more data in an Rbsp before the `rbp_trailing_bits()` syntax structure, the return value of `more_rbsp_data()` is equal to TRUE.
 - Otherwise, the return value of `more_rbsp_data()` is equal to FALSE.

The method for enabling determination of whether there is more data in the Rbsp is specified by the application (or in Annex B for applications that use the byte stream format).

`more_rbsp_trailing_data()` is specified as follows:

- If there is more data in an Rbsp, the return value of `more_rbsp_trailing_data()` is equal to TRUE.
- Otherwise, the return value of `more_rbsp_trailing_data()` is equal to FALSE.

`next_bits(n)` provides the next bits in the bitstream for comparison purposes, without advancing the bitstream pointer. It provides a look at the next n bits in the bitstream with n being its argument. When used within the byte stream format as specified in Annex B and fewer than n bits remain within the byte stream, `next_bits(n)` returns a value of 0.

`payload_extension_present()` is specified as follows:

- If the current position in the `sei_payload()` syntax structure is not the position of the last (least significant, right-most) bit that is equal to 1 that is less than $8 * \text{payloadSize}$ bits from the beginning of the syntax structure (i.e., the position of the `payload_bit_equal_to_one` syntax element), the return value of `payload_extension_present()` is equal to TRUE.
- Otherwise, the return value of `payload_extension_present()` is equal to FALSE.

`pic_layer_id(picX)` returns the value of the `nuh_layer_id` of the VCL NAL units in the picture `picX`.

`read_bits(n)` reads the next n bits from the bitstream and advances the bitstream pointer by n bit positions. When n is equal to 0, `read_bits(n)` is specified to return a value equal to 0 and to not advance the bitstream pointer.

The following descriptors specify the parsing process of each syntax element:

- `ae(v)`: context-adaptive arithmetic entropy-coded syntax element. The parsing process for this descriptor is specified in clause 9.3.
- `b(8)`: byte having any pattern of bit string (8 bits). The parsing process for this descriptor is specified by the return value of the function `read_bits(8)`.
- `f(n)`: fixed-pattern bit string using n bits written (from left to right) with the left bit first. The parsing process for this descriptor is specified by the return value of the function `read_bits(n)`.
- `i(n)`: signed integer using n bits. When n is "v" in the syntax table, the number of bits varies in a manner dependent on the value of other syntax elements. The parsing process for this descriptor is specified by the return value of the function `read_bits(n)` interpreted as a two's complement integer representation with most significant bit written first.
- `se(v)`: signed integer 0-th order Exp-Golomb-coded syntax element with the left bit first. The parsing process for this descriptor is specified in clause 9.2.

- **st(v)**: null-terminated string encoded as universal coded character set (UCS) transmission format-8 (UTF-8) characters as specified in ISO/IEC 10646. The parsing process is specified as follows: **st(v)** begins at a byte-aligned position in the bitstream and reads and returns a series of bytes from the bitstream, beginning at the current position and continuing up to but not including the next byte-aligned byte that is equal to 0x00, and advances the bitstream pointer by $(\text{stringLength} + 1) * 8$ bit positions, where **stringLength** is equal to the number of bytes returned.

NOTE – The **st(v)** syntax descriptor is only used in this Specification when the current position in the bitstream is a byte-aligned position.

- **u(n)**: unsigned integer using **n** bits. When **n** is "v" in the syntax table, the number of bits varies in a manner dependent on the value of other syntax elements. The parsing process for this descriptor is specified by the return value of the function **read_bits(n)** interpreted as a binary representation of an unsigned integer with most significant bit written first.
- **ue(v)**: unsigned integer 0-th order Exp-Golomb-coded syntax element with the left bit first. The parsing process for this descriptor is specified in clause 9.2.

7.3 Syntax in tabular form

7.3.1 NAL unit syntax

7.3.1.1 General NAL unit syntax

nal_unit(NumBytesInNalUnit) {	Descriptor
nal_unit_header()	
NumBytesInRbsp = 0	
for(i = 2; i < NumBytesInNalUnit; i++)	
if(i + 2 < NumBytesInNalUnit && next_bits(24) == 0x000003) {	
rbsp_byte [NumBytesInRbsp++]	b(8)
rbsp_byte [NumBytesInRbsp++]	b(8)
i += 2	
emulation_prevention_three_byte /* equal to 0x03 */	f(8)
} else	
rbsp_byte [NumBytesInRbsp++]	b(8)
}	

7.3.1.2 NAL unit header syntax

nal_unit_header() {	Descriptor
forbidden_zero_bit	f(1)
nal_unit_type	u(6)
nuh_layer_id	u(6)
nuh_temporal_id_plus1	u(3)
}	

7.3.2 Raw byte sequence payloads, trailing bits and byte alignment syntax

7.3.2.1 Video parameter set RBSP syntax

video_parameter_set_rbsp() {	Descriptor
vps_video_parameter_set_id	u(4)
vps_base_layer_internal_flag	u(1)
vps_base_layer_available_flag	u(1)
vps_max_layers_minus1	u(6)
vps_max_sub_layers_minus1	u(3)
vps_temporal_id_nesting_flag	u(1)
vps_reserved_0xffff_16bits	u(16)
profile_tier_level(1, vps_max_sub_layers_minus1)	
vps_sub_layer_ordering_info_present_flag	u(1)
for(i = (vps_sub_layer_ordering_info_present_flag ? 0 : vps_max_sub_layers_minus1); i <= vps_max_sub_layers_minus1; i++) {	
vps_max_dec_pic_buffering_minus1[i]	ue(v)
vps_max_num_reorder_pics[i]	ue(v)
vps_max_latency_increase_plus1[i]	ue(v)
}	
vps_max_layer_id	u(6)
vps_num_layer_sets_minus1	ue(v)
for(i = 1; i <= vps_num_layer_sets_minus1; i++)	
for(j = 0; j <= vps_max_layer_id; j++)	
layer_id_included_flag[i][j]	u(1)
vps_timing_info_present_flag	u(1)
if(vps_timing_info_present_flag) {	
vps_num_units_in_tick	u(32)
vps_time_scale	u(32)
vps_poc_proportional_to_timing_flag	u(1)
if(vps_poc_proportional_to_timing_flag)	
vps_num_ticks_poc_diff_one_minus1	ue(v)
vps_num_hrd_parameters	ue(v)
for(i = 0; i < vps_num_hrd_parameters; i++) {	
hrd_layer_set_idx[i]	ue(v)
if(i > 0)	
cprms_present_flag[i]	u(1)
hrd_parameters(cprms_present_flag[i], vps_max_sub_layers_minus1)	
}	
}	
vps_extension_flag	u(1)
if(vps_extension_flag)	
while(more_rbsp_data())	
vps_extension_data_flag	u(1)
rbsp_trailing_bits()	
}	

7.3.2.2 Sequence parameter set RBSP syntax

7.3.2.2.1 General sequence parameter set RBSP syntax

seq_parameter_set_rbsp() {	Descriptor
sps_video_parameter_set_id	u(4)
sps_max_sub_layers_minus1	u(3)
sps_temporal_id_nesting_flag	u(1)
profile_tier_level(1, sps_max_sub_layers_minus1)	
sps_seq_parameter_set_id	ue(v)
chroma_format_idc	ue(v)
if(chroma_format_idc == 3)	
separate_colour_plane_flag	u(1)
pic_width_in_luma_samples	ue(v)
pic_height_in_luma_samples	ue(v)
conformance_window_flag	u(1)
if(conformance_window_flag) {	
conf_win_left_offset	ue(v)
conf_win_right_offset	ue(v)
conf_win_top_offset	ue(v)
conf_win_bottom_offset	ue(v)
}	
bit_depth_luma_minus8	ue(v)
bit_depth_chroma_minus8	ue(v)
log2_max_pic_order_cnt_lsb_minus4	ue(v)
sps_sub_layer_ordering_info_present_flag	u(1)
for(i = (sps_sub_layer_ordering_info_present_flag ? 0 : sps_max_sub_layers_minus1); i <= sps_max_sub_layers_minus1; i++) {	
sps_max_dec_pic_buffering_minus1[i]	ue(v)
sps_max_num_reorder_pics[i]	ue(v)
sps_max_latency_increase_plus1[i]	ue(v)
}	
log2_min_luma_coding_block_size_minus3	ue(v)
log2_diff_max_min_luma_coding_block_size	ue(v)
log2_min_luma_transform_block_size_minus2	ue(v)
log2_diff_max_min_luma_transform_block_size	ue(v)
max_transform_hierarchy_depth_inter	ue(v)
max_transform_hierarchy_depth_intra	ue(v)
scaling_list_enabled_flag	u(1)
if(scaling_list_enabled_flag) {	
sps_scaling_list_data_present_flag	u(1)
if(sps_scaling_list_data_present_flag)	
scaling_list_data()	
}	
amp_enabled_flag	u(1)
sample_adaptive_offset_enabled_flag	u(1)
pcm_enabled_flag	u(1)
if(pcm_enabled_flag) {	
pcm_sample_bit_depth_luma_minus1	u(4)
pcm_sample_bit_depth_chroma_minus1	u(4)

log2_min_pcm_luma_coding_block_size_minus3	ue(v)
log2_diff_max_min_pcm_luma_coding_block_size	ue(v)
pcm_loop_filter_disabled_flag	u(1)
}	
num_short_term_ref_pic_sets	ue(v)
for(i = 0; i < num_short_term_ref_pic_sets; i++)	
st_ref_pic_set(i)	
long_term_ref_pics_present_flag	u(1)
if(long_term_ref_pics_present_flag) {	
num_long_term_ref_pics_sps	ue(v)
for(i = 0; i < num_long_term_ref_pics_sps; i++) {	
lt_ref_pic_poc_lsb_sps[i]	u(v)
used_by_curr_pic_lt_sps_flag[i]	u(1)
}	
}	
sps_temporal_mvp_enabled_flag	u(1)
strong_intra_smoothing_enabled_flag	u(1)
vui_parameters_present_flag	u(1)
if(vui_parameters_present_flag)	
vui_parameters()	
sps_extension_present_flag	u(1)
if(sps_extension_present_flag) {	
sps_range_extension_flag	u(1)
sps_multilayer_extension_flag	u(1)
sps_3d_extension_flag	u(1)
sps_scc_extension_flag	u(1)
sps_extension_4bits	u(4)
}	
if(sps_range_extension_flag)	
sps_range_extension()	
if(sps_multilayer_extension_flag)	
sps_multilayer_extension() /* specified in Annex F */	
if(sps_3d_extension_flag)	
sps_3d_extension() /* specified in Annex I */	
if(sps_scc_extension_flag)	
sps_scc_extension()	
if(sps_extension_4bits)	
while(more_rbsp_data())	
sps_extension_data_flag	u(1)
rbsp_trailing_bits()	
}	

7.3.2.2.2 Sequence parameter set range extension syntax

sps_range_extension() {	Descriptor
transform_skip_rotation_enabled_flag	u(1)
transform_skip_context_enabled_flag	u(1)
implicit_rdpem_enabled_flag	u(1)
explicit_rdpem_enabled_flag	u(1)
extended_precision_processing_flag	u(1)
intra_smoothing_disabled_flag	u(1)
high_precision_offsets_enabled_flag	u(1)
persistent_rice_adaptation_enabled_flag	u(1)
cabac_bypass_alignment_enabled_flag	u(1)
}	

7.3.2.2.3 Sequence parameter set screen content coding extension syntax

sps_scc_extension() {	Descriptor
sps_curr_pic_ref_enabled_flag	u(1)
palette_mode_enabled_flag	u(1)
if(palette_mode_enabled_flag) {	
palette_max_size	ue(v)
delta_palette_max_predictor_size	ue(v)
sps_palette_predictor_initializers_present_flag	u(1)
if(sps_palette_predictor_initializers_present_flag) {	
sps_num_palette_predictor_initializers_minus1	ue(v)
numComps = (chroma_format_idc == 0) ? 1 : 3	
for(comp = 0; comp < numComps; comp++)	
for(i = 0; i <= sps_num_palette_predictor_initializers_minus1; i++)	
sps_palette_predictor_initializer[comp][i]	u(v)
}	
}	
motion_vector_resolution_control_idc	u(2)
intra_boundary_filtering_disabled_flag	u(1)
}	

7.3.2.3 Picture parameter set RBSP syntax

7.3.2.3.1 General picture parameter set RBSP syntax

pic_parameter_set_rbsp() {	Descriptor
pps_pic_parameter_set_id	ue(v)
pps_seq_parameter_set_id	ue(v)
dependent_slice_segments_enabled_flag	u(1)
output_flag_present_flag	u(1)
num_extra_slice_header_bits	u(3)
sign_data_hiding_enabled_flag	u(1)
cabac_init_present_flag	u(1)

num_ref_idx_l0_default_active_minus1	ue(v)
num_ref_idx_l1_default_active_minus1	ue(v)
init_qp_minus26	se(v)
constrained_intra_pred_flag	u(1)
transform_skip_enabled_flag	u(1)
cu_qp_delta_enabled_flag	u(1)
if(cu_qp_delta_enabled_flag)	
diff_cu_qp_delta_depth	ue(v)
pps_cb_qp_offset	se(v)
pps_cr_qp_offset	se(v)
pps_slice_chroma_qp_offsets_present_flag	u(1)
weighted_pred_flag	u(1)
weighted_bipred_flag	u(1)
transquant_bypass_enabled_flag	u(1)
tiles_enabled_flag	u(1)
entropy_coding_sync_enabled_flag	u(1)
if(tiles_enabled_flag) {	
num_tile_columns_minus1	ue(v)
num_tile_rows_minus1	ue(v)
uniform_spacing_flag	u(1)
if(!uniform_spacing_flag) {	
for(i = 0; i < num_tile_columns_minus1; i++)	
column_width_minus1[i]	ue(v)
for(i = 0; i < num_tile_rows_minus1; i++)	
row_height_minus1[i]	ue(v)
}	
loop_filter_across_tiles_enabled_flag	u(1)
}	
pps_loop_filter_across_slices_enabled_flag	u(1)
deblocking_filter_control_present_flag	u(1)
if(deblocking_filter_control_present_flag) {	
deblocking_filter_override_enabled_flag	u(1)
pps_deblocking_filter_disabled_flag	u(1)
if(!pps_deblocking_filter_disabled_flag) {	
pps_beta_offset_div2	se(v)
pps_tc_offset_div2	se(v)
}	
}	
pps_scaling_list_data_present_flag	u(1)
if(pps_scaling_list_data_present_flag)	
scaling_list_data()	
lists_modification_present_flag	u(1)
log2_parallel_merge_level_minus2	ue(v)
slice_segment_header_extension_present_flag	u(1)
pps_extension_present_flag	u(1)
if(pps_extension_present_flag) {	
pps_range_extension_flag	u(1)
pps_multilayer_extension_flag	u(1)

pps_3d_extension_flag	u(1)
pps_scc_extension_flag	u(1)
pps_extension_4bits	u(4)
}	
if(pps_range_extension_flag)	
pps_range_extension()	
if(pps_multilayer_extension_flag)	
pps_multilayer_extension() /* specified in Annex F */	
if(pps_3d_extension_flag)	
pps_3d_extension() /* specified in Annex I */	
if(pps_scc_extension_flag)	
pps_scc_extension()	
if(pps_extension_4bits)	
while(more_rbsp_data())	
pps_extension_data_flag	u(1)
rbsp_trailing_bits()	
}	

7.3.2.3.2 Picture parameter set range extension syntax

pps_range_extension() {	Descriptor
if(transform_skip_enabled_flag)	
log2_max_transform_skip_block_size_minus2	ue(v)
cross_component_prediction_enabled_flag	u(1)
chroma_qp_offset_list_enabled_flag	u(1)
if(chroma_qp_offset_list_enabled_flag) {	
diff_cu_chroma_qp_offset_depth	ue(v)
chroma_qp_offset_list_len_minus1	ue(v)
for(i = 0; i <= chroma_qp_offset_list_len_minus1; i++) {	
cb_qp_offset_list[i]	se(v)
cr_qp_offset_list[i]	se(v)
}	
}	
log2_sao_offset_scale_luma	ue(v)
log2_sao_offset_scale_chroma	ue(v)
}	

7.3.2.3.3 Picture parameter set screen content coding extension syntax

pps_scc_extension() {	Descriptor
pps_curr_pic_ref_enabled_flag	u(1)
residual_adaptive_colour_transform_enabled_flag	u(1)
if(residual_adaptive_colour_transform_enabled_flag) {	
pps_slice_act_qp_offsets_present_flag	u(1)
pps_act_y_qp_offset_plus5	se(v)
pps_act_cb_qp_offset_plus5	se(v)
pps_act_cr_qp_offset_plus3	se(v)
}	
pps_palette_predictor_initializers_present_flag	u(1)
if(pps_palette_predictor_initializers_present_flag) {	
pps_num_palette_predictor_initializers	ue(v)
if(pps_num_palette_predictor_initializers > 0) {	
monochrome_palette_flag	u(1)
luma_bit_depth_entry_minus8	ue(v)
if(!monochrome_palette_flag)	
chroma_bit_depth_entry_minus8	ue(v)
numComps = monochrome_palette_flag ? 1 : 3	
for(comp = 0; comp < numComps; comp++)	
for(i = 0; i < pps_num_palette_predictor_initializers; i++)	
pps_palette_predictor_initializer[comp][i]	u(v)
}	
}	
}	

7.3.2.4 Supplemental enhancement information RBSP syntax

sei_rbsp() {	Descriptor
do	
sei_message()	
while(more_rbsp_data())	
rbp_trailing_bits()	
}	

7.3.2.5 Access unit delimiter RBSP syntax

access_unit_delimiter_rbsp() {	Descriptor
pic_type	u(3)
rbp_trailing_bits()	
}	

7.3.2.6 End of sequence RBSP syntax

end_of_seq_rbsp() {	Descriptor
}	

7.3.2.7 End of bitstream RBSP syntax

end_of_bitstream_rbsp() {	Descriptor
}	

7.3.2.8 Filler data RBSP syntax

filler_data_rbsp() {	Descriptor
while(next_bits(8) == 0xFF)	
ff_byte /* equal to 0xFF */	f(8)
rbsp_trailing_bits()	
}	

7.3.2.9 Slice segment layer RBSP syntax

slice_segment_layer_rbsp() {	Descriptor
slice_segment_header()	
slice_segment_data()	
rbsp_slice_segment_trailing_bits()	
}	

7.3.2.10 RBSP slice segment trailing bits syntax

rbsp_slice_segment_trailing_bits() {	Descriptor
rbsp_trailing_bits()	
while(more_rbsp_trailing_data())	
cabac_zero_word /* equal to 0x0000 */	f(16)
}	

7.3.2.11 RBSP trailing bits syntax

rbsp_trailing_bits() {	Descriptor
rbsp_stop_one_bit /* equal to 1 */	f(1)
while(!byte_aligned())	
rbsp_alignment_zero_bit /* equal to 0 */	f(1)
}	

7.3.2.12 Byte alignment syntax

byte_alignment() {	Descriptor
alignment_bit_equal_to_one /* equal to 1 */	f(1)
while(!byte_aligned())	
alignment_bit_equal_to_zero /* equal to 0 */	f(1)
}	

7.3.3 Profile, tier and level syntax

profile_tier_level(profilePresentFlag, maxNumSubLayersMinus1) {	Descriptor
if(profilePresentFlag) {	
general_profile_space	u(2)
general_tier_flag	u(1)
general_profile_idc	u(5)
for(j = 0; j < 32; j++)	
general_profile_compatibility_flag[j]	u(1)
general_progressive_source_flag	u(1)
general_interlaced_source_flag	u(1)
general_non_packed_constraint_flag	u(1)
general_frame_only_constraint_flag	u(1)
if(general_profile_idc == 4 general_profile_compatibility_flag[4] general_profile_idc == 5 general_profile_compatibility_flag[5] general_profile_idc == 6 general_profile_compatibility_flag[6] general_profile_idc == 7 general_profile_compatibility_flag[7] general_profile_idc == 8 general_profile_compatibility_flag[8] general_profile_idc == 9 general_profile_compatibility_flag[9] general_profile_idc == 10 general_profile_compatibility_flag[10] general_profile_idc == 11 general_profile_compatibility_flag[11]) { /* The number of bits in this syntax structure is not affected by this condition */	
general_max_12bit_constraint_flag	u(1)
general_max_10bit_constraint_flag	u(1)
general_max_8bit_constraint_flag	u(1)
general_max_422chroma_constraint_flag	u(1)
general_max_420chroma_constraint_flag	u(1)
general_max_monochrome_constraint_flag	u(1)
general_intra_constraint_flag	u(1)
general_one_picture_only_constraint_flag	u(1)
general_lower_bit_rate_constraint_flag	u(1)
if(general_profile_idc == 5 general_profile_compatibility_flag[5] general_profile_idc == 9 general_profile_compatibility_flag[9] general_profile_idc == 10 general_profile_compatibility_flag[10] general_profile_idc == 11 general_profile_compatibility_flag[11]) {	
general_max_14bit_constraint_flag	u(1)
general_reserved_zero_33bits	u(33)
} else	
general_reserved_zero_34bits	u(34)
} else if(general_profile_idc == 2 general_profile_compatibility_flag[2]) {	
general_reserved_zero_7bits	u(7)
general_one_picture_only_constraint_flag	u(1)

general_reserved_zero_35bits	u(35)
} else	
general_reserved_zero_43bits	u(43)
<pre> if(general_profile_idc == 1 general_profile_compatibility_flag[1] general_profile_idc == 2 general_profile_compatibility_flag[2] general_profile_idc == 3 general_profile_compatibility_flag[3] general_profile_idc == 4 general_profile_compatibility_flag[4] general_profile_idc == 5 general_profile_compatibility_flag[5] general_profile_idc == 9 general_profile_compatibility_flag[9] general_profile_idc == 11 general_profile_compatibility_flag[11]) /* The number of bits in this syntax structure is not affected by this condition */ </pre>	
general_inbld_flag	u(1)
else	
general_reserved_zero_bit	u(1)
}	
general_level_idc	u(8)
for(i = 0; i < maxNumSubLayersMinus1; i++) {	
sub_layer_profile_present_flag[i]	u(1)
sub_layer_level_present_flag[i]	u(1)
}	
if(maxNumSubLayersMinus1 > 0)	
for(i = maxNumSubLayersMinus1; i < 8; i++)	
reserved_zero_2bits[i]	u(2)
for(i = 0; i < maxNumSubLayersMinus1; i++) {	
if(sub_layer_profile_present_flag[i]) {	
sub_layer_profile_space[i]	u(2)
sub_layer_tier_flag[i]	u(1)
sub_layer_profile_idc[i]	u(5)
for(j = 0; j < 32; j++)	
sub_layer_profile_compatibility_flag[i][j]	u(1)
sub_layer_progressive_source_flag[i]	u(1)
sub_layer_interlaced_source_flag[i]	u(1)
sub_layer_non_packed_constraint_flag[i]	u(1)
sub_layer_frame_only_constraint_flag[i]	u(1)
<pre> if(sub_layer_profile_idc[i] == 4 sub_layer_profile_compatibility_flag[i][4] sub_layer_profile_idc[i] == 5 sub_layer_profile_compatibility_flag[i][5] sub_layer_profile_idc[i] == 6 sub_layer_profile_compatibility_flag[i][6] sub_layer_profile_idc[i] == 7 sub_layer_profile_compatibility_flag[i][7] sub_layer_profile_idc[i] == 8 sub_layer_profile_compatibility_flag[i][8] sub_layer_profile_idc[i] == 9 sub_layer_profile_compatibility_flag[i][9] sub_layer_profile_idc[i] == 10 sub_layer_profile_compatibility_flag[i][10] sub_layer_profile_idc[i] == 11 sub_layer_profile_compatibility_flag[i][11]) { /* The number of bits in this syntax structure is not affected by this condition */ </pre>	
sub_layer_max_12bit_constraint_flag[i]	u(1)
sub_layer_max_10bit_constraint_flag[i]	u(1)
sub_layer_max_8bit_constraint_flag[i]	u(1)

sub_layer_max_422chroma_constraint_flag[i]	u(1)
sub_layer_max_420chroma_constraint_flag[i]	u(1)
sub_layer_max_monochrome_constraint_flag[i]	u(1)
sub_layer_intra_constraint_flag[i]	u(1)
sub_layer_one_picture_only_constraint_flag[i]	u(1)
sub_layer_lower_bit_rate_constraint_flag[i]	u(1)
if(sub_layer_profile_idc[i] == 5 sub_layer_profile_compatibility_flag[i][5] sub_layer_profile_idc[i] == 9 sub_layer_profile_compatibility_flag[i][9] sub_layer_profile_idc[i] == 10 sub_layer_profile_compatibility_flag[i][10] sub_layer_profile_idc[i] == 11 sub_layer_profile_compatibility_flag[i][11]) {	
sub_layer_max_14bit_constraint_flag[i]	u(1)
sub_layer_reserved_zero_33bits[i]	u(33)
} else	
sub_layer_reserved_zero_34bits[i]	u(34)
} else if(sub_layer_profile_idc[i] == 2 sub_layer_profile_compatibility_flag[i][2]) {	
sub_layer_reserved_zero_7bits[i]	u(7)
sub_layer_one_picture_only_constraint_flag[i]	u(1)
sub_layer_reserved_zero_35bits[i]	u(35)
} else	
sub_layer_reserved_zero_43bits[i]	u(43)
if(sub_layer_profile_idc[i] == 1 sub_layer_profile_compatibility_flag[i][1] sub_layer_profile_idc[i] == 2 sub_layer_profile_compatibility_flag[i][2] sub_layer_profile_idc[i] == 3 sub_layer_profile_compatibility_flag[i][3] sub_layer_profile_idc[i] == 4 sub_layer_profile_compatibility_flag[i][4] sub_layer_profile_idc[i] == 5 sub_layer_profile_compatibility_flag[i][5] sub_layer_profile_idc[i] == 9 sub_layer_profile_compatibility_flag[i][9] sub_layer_profile_idc[i] == 11 sub_layer_profile_compatibility_flag[i][11]) /* The number of bits in this syntax structure is not affected by this condition */	
sub_layer_inbld_flag[i]	u(1)
else	
sub_layer_reserved_zero_bit[i]	u(1)
}	
if(sub_layer_level_present_flag[i])	
sub_layer_level_idc[i]	u(8)
}	
}	

7.3.4 Scaling list data syntax

scaling_list_data() {	Descriptor
for(sizeId = 0; sizeId < 4; sizeId++)	
for(matrixId = 0; matrixId < 6; matrixId += (sizeId == 3) ? 3 : 1) {	
scaling_list_pred_mode_flag [sizeId][matrixId]	u(1)
if(!scaling_list_pred_mode_flag[sizeId][matrixId])	
scaling_list_pred_matrix_id_delta [sizeId][matrixId]	ue(v)
else {	
nextCoef = 8	
coefNum = Min(64, (1 << (4 + (sizeId << 1))))	
if(sizeId > 1) {	
scaling_list_dc_coef_minus8 [sizeId - 2][matrixId]	se(v)
nextCoef = scaling_list_dc_coef_minus8[sizeId - 2][matrixId] + 8	
}	
for(i = 0; i < coefNum; i++) {	
scaling_list_delta_coef	se(v)
nextCoef = (nextCoef + scaling_list_delta_coef + 256) % 256	
ScalingList[sizeId][matrixId][i] = nextCoef	
}	
}	
}	
}	

7.3.5 Supplemental enhancement information message syntax

sei_message() {	Descriptor
payloadType = 0	
while(next_bits(8) == 0xFF) {	
ff_byte /* equal to 0xFF */	f(8)
payloadType += 255	
}	
last_payload_type_byte	u(8)
payloadType += last_payload_type_byte	
payloadSize = 0	
while(next_bits(8) == 0xFF) {	
ff_byte /* equal to 0xFF */	f(8)
payloadSize += 255	
}	
last_payload_size_byte	u(8)
payloadSize += last_payload_size_byte	
sei_payload(payloadType, payloadSize)	
}	

7.3.6 Slice segment header syntax

7.3.6.1 General slice segment header syntax

slice_segment_header() {	Descriptor
first_slice_segment_in_pic_flag	u(1)
if(nal_unit_type >= BLA_W_LP && nal_unit_type <= RSV_IRAP_VCL23)	
no_output_of_prior_pics_flag	u(1)
slice_pic_parameter_set_id	ue(v)
if(!first_slice_segment_in_pic_flag) {	
if(dependent_slice_segments_enabled_flag)	
dependent_slice_segment_flag	u(1)
slice_segment_address	u(v)
}	
CuQpDeltaVal = 0	
if(!dependent_slice_segment_flag) {	
for(i = 0; i < num_extra_slice_header_bits; i++)	
slice_reserved_flag[i]	u(1)
slice_type	ue(v)
if(output_flag_present_flag)	
pic_output_flag	u(1)
if(separate_colour_plane_flag == 1)	
colour_plane_id	u(2)
if(nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP) {	
slice_pic_order_cnt_lsb	u(v)
short_term_ref_pic_set_sps_flag	u(1)
if(!short_term_ref_pic_set_sps_flag)	
st_ref_pic_set(num_short_term_ref_pic_sets)	
else if(num_short_term_ref_pic_sets > 1)	
short_term_ref_pic_set_idx	u(v)
if(long_term_ref_pics_present_flag) {	
if(num_long_term_ref_pics_sps > 0)	
num_long_term_sps	ue(v)
num_long_term_pics	ue(v)
for(i = 0; i < num_long_term_sps + num_long_term_pics; i++) {	
if(i < num_long_term_sps) {	
if(num_long_term_ref_pics_sps > 1)	
lt_idx_sps[i]	u(v)
} else {	
poc_lsb_lt[i]	u(v)
used_by_curr_pic_lt_flag[i]	u(1)
}	
delta_poc_msb_present_flag[i]	u(1)
if(delta_poc_msb_present_flag[i])	
delta_poc_msb_cycle_lt[i]	ue(v)
}	
}	
if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)

}	
if(sample_adaptive_offset_enabled_flag) {	
slice_sao_luma_flag	u(1)
if(ChromaArrayType != 0)	
slice_sao_chroma_flag	u(1)
}	
if(slice_type == P slice_type == B) {	
num_ref_idx_active_override_flag	u(1)
if(num_ref_idx_active_override_flag) {	
num_ref_idx_l0_active_minus1	ue(v)
if(slice_type == B)	
num_ref_idx_l1_active_minus1	ue(v)
}	
if(lists_modification_present_flag && NumPicTotalCurr > 1)	
ref_pic_lists_modification()	
if(slice_type == B)	
mvd_l1_zero_flag	u(1)
if(cabac_init_present_flag)	
cabac_init_flag	u(1)
if(slice_temporal_mvp_enabled_flag) {	
if(slice_type == B)	
collocated_from_l0_flag	u(1)
if((collocated_from_l0_flag && num_ref_idx_l0_active_minus1 > 0) (!collocated_from_l0_flag && num_ref_idx_l1_active_minus1 > 0))	
collocated_ref_idx	ue(v)
}	
if((weighted_pred_flag && slice_type == P) (weighted_bipred_flag && slice_type == B))	
pred_weight_table()	
five_minus_max_num_merge_cand	ue(v)
if(motion_vector_resolution_control_idc == 2)	
use_integer_mv_flag	u(1)
}	
slice_qp_delta	se(v)
if(pps_slice_chroma_qp_offsets_present_flag) {	
slice_cb_qp_offset	se(v)
slice_cr_qp_offset	se(v)
}	
if(pps_slice_act_qp_offsets_present_flag) {	
slice_act_y_qp_offset	se(v)
slice_act_cb_qp_offset	se(v)
slice_act_cr_qp_offset	se(v)
}	
if(chroma_qp_offset_list_enabled_flag)	
cu_chroma_qp_offset_enabled_flag	u(1)
if(deblocking_filter_override_enabled_flag)	
deblocking_filter_override_flag	u(1)

if(deblocking_filter_override_flag) {	
slice_deblocking_filter_disabled_flag	u(1)
if(!slice_deblocking_filter_disabled_flag) {	
slice_beta_offset_div2	se(v)
slice_tc_offset_div2	se(v)
}	
}	
if(pps_loop_filter_across_slices_enabled_flag && (slice_sao_luma_flag slice_sao_chroma_flag !slice_deblocking_filter_disabled_flag))	
slice_loop_filter_across_slices_enabled_flag	u(1)
}	
if(tiles_enabled_flag entropy_coding_sync_enabled_flag) {	
num_entry_point_offsets	ue(v)
if(num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset_minus1[i]	u(v)
}	
}	
if(slice_segment_header_extension_present_flag) {	
slice_segment_header_extension_length	ue(v)
for(i = 0; i < slice_segment_header_extension_length; i++)	
slice_segment_header_extension_data_byte[i]	u(8)
}	
byte_alignment()	
}	

7.3.6.2 Reference picture list modification syntax

ref_pic_lists_modification() {	Descriptor
ref_pic_list_modification_flag_l0	u(1)
if(ref_pic_list_modification_flag_l0)	
for(i = 0; i <= num_ref_idx_l0_active_minus1; i++)	
list_entry_l0[i]	u(v)
if(slice_type == B) {	
ref_pic_list_modification_flag_l1	u(1)
if(ref_pic_list_modification_flag_l1)	
for(i = 0; i <= num_ref_idx_l1_active_minus1; i++)	
list_entry_l1[i]	u(v)
}	
}	

7.3.6.3 Weighted prediction parameters syntax

	Descriptor
pred_weight_table() {	
luma_log2_weight_denom	ue(v)
if(ChromaArrayType != 0)	
delta_chroma_log2_weight_denom	se(v)
for(i = 0; i <= num_ref_idx_l0_active_minus1; i++)	
if((pic_layer_id(RefPicList0[i]) != nuh_layer_id) (PicOrderCnt(RefPicList0[i]) != PicOrderCnt(CurrPic)))	
luma_weight_l0_flag[i]	u(1)
if(ChromaArrayType != 0)	
for(i = 0; i <= num_ref_idx_l0_active_minus1; i++)	
if((pic_layer_id(RefPicList0[i]) != nuh_layer_id) (PicOrderCnt(RefPicList0[i]) != PicOrderCnt(CurrPic)))	
chroma_weight_l0_flag[i]	u(1)
for(i = 0; i <= num_ref_idx_l0_active_minus1; i++) {	
if(luma_weight_l0_flag[i]) {	
delta_luma_weight_l0[i]	se(v)
luma_offset_l0[i]	se(v)
}	
if(chroma_weight_l0_flag[i])	
for(j = 0; j < 2; j++) {	
delta_chroma_weight_l0[i][j]	se(v)
delta_chroma_offset_l0[i][j]	se(v)
}	
}	
if(slice_type == B) {	
for(i = 0; i <= num_ref_idx_l1_active_minus1; i++)	
if((pic_layer_id(RefPicList0[i]) != nuh_layer_id) (PicOrderCnt(RefPicList1[i]) != PicOrderCnt(CurrPic)))	
luma_weight_l1_flag[i]	u(1)
if(ChromaArrayType != 0)	
for(i = 0; i <= num_ref_idx_l1_active_minus1; i++)	
if((pic_layer_id(RefPicList0[i]) != nuh_layer_id) (PicOrderCnt(RefPicList1[i]) != PicOrderCnt(CurrPic)))	
chroma_weight_l1_flag[i]	u(1)
for(i = 0; i <= num_ref_idx_l1_active_minus1; i++) {	
if(luma_weight_l1_flag[i]) {	
delta_luma_weight_l1[i]	se(v)
luma_offset_l1[i]	se(v)
}	

if(chroma_weight_l1_flag[i])	
for(j = 0; j < 2; j++) {	
delta_chroma_weight_l1 [i][j]	se(v)
delta_chroma_offset_l1 [i][j]	se(v)
}	
}	
}	
}	

7.3.7 Short-term reference picture set syntax

	Descriptor
st_ref_pic_set(stRpsIdx) {	
if(stRpsIdx != 0)	
inter_ref_pic_set_prediction_flag	u(1)
if(inter_ref_pic_set_prediction_flag) {	
if(stRpsIdx == num_short_term_ref_pic_sets)	
delta_idx_minus1	ue(v)
delta_rps_sign	u(1)
abs_delta_rps_minus1	ue(v)
for(j = 0; j <= NumDeltaPocs[RefRpsIdx]; j++) {	
used_by_curr_pic_flag [j]	u(1)
if(!used_by_curr_pic_flag[j])	
use_delta_flag [j]	u(1)
}	
} else {	
num_negative_pics	ue(v)
num_positive_pics	ue(v)
for(i = 0; i < num_negative_pics; i++) {	
delta_poc_s0_minus1 [i]	ue(v)
used_by_curr_pic_s0_flag [i]	u(1)
}	
for(i = 0; i < num_positive_pics; i++) {	
delta_poc_s1_minus1 [i]	ue(v)
used_by_curr_pic_s1_flag [i]	u(1)
}	
}	
}	

7.3.8 Slice segment data syntax

7.3.8.1 General slice segment data syntax

slice_segment_data() {	Descriptor
do {	
coding_tree_unit()	
end_of_slice_segment_flag	ae(v)
CtbAddrInTs++	
CtbAddrInRs = CtbAddrTsToRs[CtbAddrInTs]	
if(!end_of_slice_segment_flag && ((tiles_enabled_flag && TileId[CtbAddrInTs] != TileId[CtbAddrInTs - 1]) (entropy_coding_sync_enabled_flag && (CtbAddrInRs % PicWidthInCtbsY == 0 TileId[CtbAddrInTs] != TileId[CtbAddrRsToTs[CtbAddrInRs - 1]]))))) {	
end_of_subset_one_bit /* equal to 1 */	ae(v)
byte_alignment()	
}	
} while(!end_of_slice_segment_flag)	
}	

7.3.8.2 Coding tree unit syntax

coding_tree_unit() {	Descriptor
xCtb = (CtbAddrInRs % PicWidthInCtbsY) << CtbLog2SizeY	
yCtb = (CtbAddrInRs / PicWidthInCtbsY) << CtbLog2SizeY	
if(slice_sao_luma_flag slice_sao_chroma_flag)	
sao(xCtb >> CtbLog2SizeY, yCtb >> CtbLog2SizeY)	
coding_quadtrees(xCtb, yCtb, CtbLog2SizeY, 0)	
}	

7.3.8.3 Sample adaptive offset syntax

	Descriptor
sao(rx, ry) {	
if(rx > 0) {	
leftCtbInSliceSeg = CtbAddrInRs > SliceAddrRs	
leftCtbInTile = TileId[CtbAddrInTs] == TileId[CtbAddrRsToTs[CtbAddrInRs - 1]]	
if(leftCtbInSliceSeg && leftCtbInTile)	
sao_merge_left_flag	ae(v)
}	
if(ry > 0 && !sao_merge_left_flag) {	
upCtbInSliceSeg = (CtbAddrInRs - PicWidthInCtbsY) >= SliceAddrRs	
upCtbInTile = TileId[CtbAddrInTs] == TileId[CtbAddrRsToTs[CtbAddrInRs - PicWidthInCtbsY]]	
if(upCtbInSliceSeg && upCtbInTile)	
sao_merge_up_flag	ae(v)
}	
if(!sao_merge_up_flag && !sao_merge_left_flag)	
for(cIdx = 0; cIdx < (ChromaArrayType != 0 ? 3 : 1); cIdx++)	
if((slice_sao_luma_flag && cIdx == 0) (slice_sao_chroma_flag && cIdx > 0)) {	
if(cIdx == 0)	
sao_type_idx_luma	ae(v)
else if(cIdx == 1)	
sao_type_idx_chroma	ae(v)
if(SaoTypeIdx[cIdx][rx][ry] != 0) {	
for(i = 0; i < 4; i++)	
sao_offset_abs [cIdx][rx][ry][i]	ae(v)
if(SaoTypeIdx[cIdx][rx][ry] == 1) {	
for(i = 0; i < 4; i++)	
if(sao_offset_abs[cIdx][rx][ry][i] != 0)	
sao_offset_sign [cIdx][rx][ry][i]	ae(v)
sao_band_position [cIdx][rx][ry]	ae(v)
} else {	
if(cIdx == 0)	
sao_eo_class_luma	ae(v)
if(cIdx == 1)	
sao_eo_class_chroma	ae(v)
}	
}	
}	
}	
}	

7.3.8.4 Coding quadtree syntax

	Descriptor
<code>coding_quadtree(x0, y0, log2CbSize, cqtDepth) {</code>	
<code> if(x0 + (1 << log2CbSize) <= pic_width_in_luma_samples && y0 + (1 << log2CbSize) <= pic_height_in_luma_samples && log2CbSize > MinCbLog2SizeY)</code>	
<code> split_cu_flag[x0][y0]</code>	<code>ae(v)</code>
<code> if(cu_qp_delta_enabled_flag && log2CbSize >= Log2MinCuQpDeltaSize) {</code>	
<code> IsCuQpDeltaCoded = 0</code>	
<code> CuQpDeltaVal = 0</code>	
<code> }</code>	
<code> if(cu_chroma_qp_offset_enabled_flag && log2CbSize >= Log2MinCuChromaQpOffsetSize)</code>	
<code> IsCuChromaQpOffsetCoded = 0</code>	
<code> if(split_cu_flag[x0][y0]) {</code>	
<code> x1 = x0 + (1 << (log2CbSize - 1))</code>	
<code> y1 = y0 + (1 << (log2CbSize - 1))</code>	
<code> coding_quadtree(x0, y0, log2CbSize - 1, cqtDepth + 1)</code>	
<code> if(x1 < pic_width_in_luma_samples)</code>	
<code> coding_quadtree(x1, y0, log2CbSize - 1, cqtDepth + 1)</code>	
<code> if(y1 < pic_height_in_luma_samples)</code>	
<code> coding_quadtree(x0, y1, log2CbSize - 1, cqtDepth + 1)</code>	
<code> if(x1 < pic_width_in_luma_samples && y1 < pic_height_in_luma_samples)</code>	
<code> coding_quadtree(x1, y1, log2CbSize - 1, cqtDepth + 1)</code>	
<code> } else</code>	
<code> coding_unit(x0, y0, log2CbSize)</code>	
<code>}</code>	

7.3.8.5 Coding unit syntax

	Descriptor
<code>coding_unit(x0, y0, log2CbSize) {</code>	
<code> if(transquant_bypass_enabled_flag)</code>	
<code> cu_transquant_bypass_flag</code>	<code>ae(v)</code>
<code> if(slice_type != I)</code>	
<code> cu_skip_flag[x0][y0]</code>	<code>ae(v)</code>
<code> nCbs = (1 << log2CbSize)</code>	
<code> if(cu_skip_flag[x0][y0])</code>	
<code> prediction_unit(x0, y0, nCbs, nCbs)</code>	
<code> else {</code>	
<code> if(slice_type != I)</code>	
<code> pred_mode_flag</code>	<code>ae(v)</code>
<code> if(palette_mode_enabled_flag && CuPredMode[x0][y0] == MODE_INTRA && log2CbSize <= MaxTbLog2SizeY)</code>	
<code> palette_mode_flag[x0][y0]</code>	<code>ae(v)</code>
<code> if(palette_mode_flag[x0][y0])</code>	
<code> palette_coding(x0, y0, nCbs)</code>	
<code> } else {</code>	
<code> if(CuPredMode[x0][y0] != MODE_INTRA log2CbSize == MinCbLog2SizeY)</code>	

part_mode	ae(v)
if(CuPredMode[x0][y0] == MODE_INTRA) {	
if(PartMode == PART_2Nx2N && pcm_enabled_flag && log2CbSize >= Log2MinIpcmCbSizeY && log2CbSize <= Log2MaxIpcmCbSizeY)	
pcm_flag [x0][y0]	ae(v)
if(pcm_flag[x0][y0]) {	
while(!byte_aligned())	
pcm_alignment_zero_bit	f(1)
pcm_sample(x0, y0, log2CbSize)	
} else {	
pbOffset = (PartMode == PART_NxN) ? (nCbS / 2) : nCbS	
for(j = 0; j < nCbS; j = j + pbOffset)	
for(i = 0; i < nCbS; i = i + pbOffset)	
prev_intra_luma_pred_flag [x0 + i][y0 + j]	ae(v)
for(j = 0; j < nCbS; j = j + pbOffset)	
for(i = 0; i < nCbS; i = i + pbOffset)	
if(prev_intra_luma_pred_flag[x0 + i][y0 + j])	
mpm_idx [x0 + i][y0 + j]	ae(v)
else	
rem_intra_luma_pred_mode [x0 + i][y0 + j]	ae(v)
if(ChromaArrayType == 3)	
for(j = 0; j < nCbS; j = j + pbOffset)	
for(i = 0; i < nCbS; i = i + pbOffset)	
intra_chroma_pred_mode [x0 + i][y0 + j]	ae(v)
else if(ChromaArrayType != 0)	
intra_chroma_pred_mode [x0][y0]	ae(v)
}	
} else {	
if(PartMode == PART_2Nx2N)	
prediction_unit(x0, y0, nCbS, nCbS)	
else if(PartMode == PART_2NxN) {	
prediction_unit(x0, y0, nCbS, nCbS / 2)	
prediction_unit(x0, y0 + (nCbS / 2), nCbS, nCbS / 2)	
} else if(PartMode == PART_Nx2N) {	
prediction_unit(x0, y0, nCbS / 2, nCbS)	
prediction_unit(x0 + (nCbS / 2), y0, nCbS / 2, nCbS)	
} else if(PartMode == PART_2NxN) {	
prediction_unit(x0, y0, nCbS, nCbS / 4)	
prediction_unit(x0, y0 + (nCbS / 4), nCbS, nCbS * 3 / 4)	
} else if(PartMode == PART_2NxN) {	
prediction_unit(x0, y0, nCbS, nCbS * 3 / 4)	
prediction_unit(x0, y0 + (nCbS * 3 / 4), nCbS, nCbS / 4)	
} else if(PartMode == PART_nLx2N) {	
prediction_unit(x0, y0, nCbS / 4, nCbS)	
prediction_unit(x0 + (nCbS / 4), y0, nCbS * 3 / 4, nCbS)	
} else if(PartMode == PART_nRx2N) {	
prediction_unit(x0, y0, nCbS * 3 / 4, nCbS)	
prediction_unit(x0 + (nCbS * 3 / 4), y0, nCbS / 4, nCbS)	

} else { /* PART_NxN */	
prediction_unit(x0, y0, nCbS / 2, nCbS / 2)	
prediction_unit(x0 + (nCbS / 2), y0, nCbS / 2, nCbS / 2)	
prediction_unit(x0, y0 + (nCbS / 2), nCbS / 2, nCbS / 2)	
prediction_unit(x0 + (nCbS / 2), y0 + (nCbS / 2), nCbS / 2, nCbS / 2)	
}	
}	
if(!pcm_flag[x0][y0]) {	
if(CuPredMode[x0][y0] != MODE_INTRA && !(PartMode == PART_2Nx2N && merge_flag[x0][y0]))	
rqt_root_cbf	ae(v)
if(rqt_root_cbf) {	
MaxTrafoDepth = (CuPredMode[x0][y0] == MODE_INTRA ? (max_transform_hierarchy_depth_intra + IntraSplitFlag) : max_transform_hierarchy_depth_inter)	
transform_tree(x0, y0, x0, y0, log2CbSize, 0, 0)	
}	
}	
}	
}	
}	
}	
}	

7.3.8.6 Prediction unit syntax

prediction_unit(x0, y0, nPbW, nPbH) {	Descriptor
if(cu_skip_flag[x0][y0]) {	
if(MaxNumMergeCand > 1)	
merge_idx [x0][y0]	ae(v)
} else { /* MODE_INTER */	
merge_flag [x0][y0]	ae(v)
if(merge_flag[x0][y0]) {	
if(MaxNumMergeCand > 1)	
merge_idx [x0][y0]	ae(v)
} else {	
if(slice_type == B)	
inter_pred_idc [x0][y0]	ae(v)
if(inter_pred_idc[x0][y0] != PRED_L1) {	
if(num_ref_idx_l0_active_minus1 > 0)	
ref_idx_l0 [x0][y0]	ae(v)
mvd_coding(x0, y0, 0)	
mvp_l0_flag [x0][y0]	ae(v)
}	
if(inter_pred_idc[x0][y0] != PRED_L0) {	
if(num_ref_idx_l1_active_minus1 > 0)	
ref_idx_l1 [x0][y0]	ae(v)

if(mvd_l1_zero_flag && inter_pred_idc[x0][y0] == PRED_BI) {	
MvdL1[x0][y0][0] = 0	
MvdL1[x0][y0][1] = 0	
} else	
mvd_coding(x0, y0, 1)	
mvp_l1_flag [x0][y0]	ae(v)
}	
}	
}	
}	

7.3.8.7 PCM sample syntax

pcm_sample(x0, y0, log2CbSize) {	Descriptor
for(i = 0; i < 1 << (log2CbSize << 1); i++)	
pcm_sample_luma [i]	u(v)
if(ChromaArrayType != 0)	
for(i = 0; i < ((2 << (log2CbSize << 1)) / (SubWidthC * SubHeightC)); i++)	
pcm_sample_chroma [i]	u(v)
}	

7.3.8.8 Transform tree syntax

transform_tree(x0, y0, xBase, yBase, log2TrafoSize, trafoDepth, blkIdx) {	Descriptor
if(log2TrafoSize <= MaxTbLog2SizeY && log2TrafoSize > MinTbLog2SizeY && trafoDepth < MaxTrafoDepth && !(IntraSplitFlag && (trafoDepth == 0)))	
split_transform_flag [x0][y0][trafoDepth]	ae(v)
if((log2TrafoSize > 2 && ChromaArrayType != 0) ChromaArrayType == 3) {	
if(trafoDepth == 0 cbf_cb[xBase][yBase][trafoDepth - 1]) {	
cbf_cb [x0][y0][trafoDepth]	ae(v)
if(ChromaArrayType == 2 && (!split_transform_flag[x0][y0][trafoDepth] log2TrafoSize == 3))	
cbf_cb [x0][y0 + (1 << (log2TrafoSize - 1))][trafoDepth]	ae(v)
}	
if(trafoDepth == 0 cbf_cr[xBase][yBase][trafoDepth - 1]) {	
cbf_cr [x0][y0][trafoDepth]	ae(v)
if(ChromaArrayType == 2 && (!split_transform_flag[x0][y0][trafoDepth] log2TrafoSize == 3))	
cbf_cr [x0][y0 + (1 << (log2TrafoSize - 1))][trafoDepth]	ae(v)
}	
}	
if(split_transform_flag[x0][y0][trafoDepth]) {	
x1 = x0 + (1 << (log2TrafoSize - 1))	
y1 = y0 + (1 << (log2TrafoSize - 1))	

transform_tree(x0, y0, x0, y0, log2TrafoSize – 1, trafoDepth + 1, 0)	
transform_tree(x1, y0, x0, y0, log2TrafoSize – 1, trafoDepth + 1, 1)	
transform_tree(x0, y1, x0, y0, log2TrafoSize – 1, trafoDepth + 1, 2)	
transform_tree(x1, y1, x0, y0, log2TrafoSize – 1, trafoDepth + 1, 3)	
} else {	
if(CuPredMode[x0][y0] == MODE_INTRA trafoDepth != 0 cbf_cb[x0][y0][trafoDepth] cbf_cr[x0][y0][trafoDepth] (ChromaArrayType == 2 && (cbf_cb[x0][y0 + (1 << (log2TrafoSize – 1))][trafoDepth] cbf_cr[x0][y0 + (1 << (log2TrafoSize – 1))][trafoDepth])))	
cbf_luma [x0][y0][trafoDepth]	ae(v)
transform_unit(x0, y0, xBase, yBase, log2TrafoSize, trafoDepth, blkIdx)	
}	
}	

7.3.8.9 Motion vector difference syntax

mvd_coding(x0, y0, refList) {	Descriptor
abs_mvd_greater0_flag [0]	ae(v)
abs_mvd_greater0_flag [1]	ae(v)
if(abs_mvd_greater0_flag[0])	
abs_mvd_greater1_flag [0]	ae(v)
if(abs_mvd_greater0_flag[1])	
abs_mvd_greater1_flag [1]	ae(v)
if(abs_mvd_greater0_flag[0]) {	
if(abs_mvd_greater1_flag[0])	
abs_mvd_minus2 [0]	ae(v)
mvd_sign_flag [0]	ae(v)
}	
if(abs_mvd_greater0_flag[1]) {	
if(abs_mvd_greater1_flag[1])	
abs_mvd_minus2 [1]	ae(v)
mvd_sign_flag [1]	ae(v)
}	
}	

7.3.8.10 Transform unit syntax

transform_unit(x0, y0, xBase, yBase, log2TrafoSize, trafoDepth, blkIdx) {	Descriptor
log2TrafoSizeC = Max(2, log2TrafoSize – (ChromaArrayType == 3 ? 0 : 1))	
cbfDepthC = trafoDepth – (ChromaArrayType != 3 && log2TrafoSize == 2 ? 1 : 0)	
xC = (ChromaArrayType != 3 && log2TrafoSize == 2) ? xBase : x0	
yC = (ChromaArrayType != 3 && log2TrafoSize == 2) ? yBase : y0	
cbfLuma = cbf_luma[x0][y0][trafoDepth]	

cbfChroma = cbf_cb[xC][yC][cbfDepthC] cbf_cr[xC][yC][cbfDepthC] (ChromaArrayType == 2 && (cbf_cb[xC][yC + (1 << log2TrafoSizeC)][cbfDepthC] cbf_cr[xC][yC + (1 << log2TrafoSizeC)][cbfDepthC]))	
if(cbfLuma cbfChroma) {	
xP = (x0 >> MinCbLog2SizeY) << MinCbLog2SizeY	
yP = (y0 >> MinCbLog2SizeY) << MinCbLog2SizeY	
nCbS = 1 << MinCbLog2SizeY	
if(residual_adaptive_colour_transform_enabled_flag && (CuPredMode[x0][y0] == MODE_INTER (PartMode == PART_2Nx2N && intra_chroma_pred_mode[x0][y0] == 4) (intra_chroma_pred_mode[xP][yP] == 4 && intra_chroma_pred_mode[xP + nCbS/2][yP] == 4 && intra_chroma_pred_mode[xP][yP + nCbS/2] == 4 && intra_chroma_pred_mode[xP + nCbS/2][yP + nCbS/2] == 4)))	
tu_residual_act_flag[x0][y0]	ae(v)
delta_qp()	
if(cbfChroma && !cu_transquant_bypass_flag)	
chroma_qp_offset()	
if(cbfLuma)	
residual_coding(x0, y0, log2TrafoSize, 0)	
if(log2TrafoSize > 2 ChromaArrayType == 3) {	
if(cross_component_prediction_enabled_flag && cbfLuma && (CuPredMode[x0][y0] == MODE_INTER intra_chroma_pred_mode[x0][y0] == 4))	
cross_comp_pred(x0, y0, 0)	
for(tIdx = 0; tIdx < (ChromaArrayType == 2 ? 2 : 1); tIdx++)	
if(cbf_cb[x0][y0 + (tIdx << log2TrafoSizeC)][trafoDepth])	
residual_coding(x0, y0 + (tIdx << log2TrafoSizeC), log2TrafoSizeC, 1)	
if(cross_component_prediction_enabled_flag && cbfLuma && (CuPredMode[x0][y0] == MODE_INTER intra_chroma_pred_mode[x0][y0] == 4))	
cross_comp_pred(x0, y0, 1)	
for(tIdx = 0; tIdx < (ChromaArrayType == 2 ? 2 : 1); tIdx++)	
if(cbf_cr[x0][y0 + (tIdx << log2TrafoSizeC)][trafoDepth])	
residual_coding(x0, y0 + (tIdx << log2TrafoSizeC), log2TrafoSizeC, 2)	
} else if(blkIdx == 3) {	
for(tIdx = 0; tIdx < (ChromaArrayType == 2 ? 2 : 1); tIdx++)	
if(cbf_cb[xBase][yBase + (tIdx << log2TrafoSizeC)][trafoDepth - 1])	
residual_coding(xBase, yBase + (tIdx << log2TrafoSizeC), log2TrafoSize, 1)	
for(tIdx = 0; tIdx < (ChromaArrayType == 2 ? 2 : 1); tIdx++)	
if(cbf_cr[xBase][yBase + (tIdx << log2TrafoSizeC)][trafoDepth - 1])	
residual_coding(xBase, yBase + (tIdx << log2TrafoSizeC), log2TrafoSize, 2)	
}	
}	
}	

7.3.8.11 Residual coding syntax

residual_coding(x0, y0, log2TrafoSize, cIdx) {	Descriptor
if(transform_skip_enabled_flag && !cu_transquant_bypass_flag && (log2TrafoSize <= Log2MaxTransformSkipSize))	
transform_skip_flag [x0][y0][cIdx]	ae(v)
if(CuPredMode[x0][y0] == MODE_INTER && explicit_rdpem_enabled_flag && (transform_skip_flag[x0][y0][cIdx] cu_transquant_bypass_flag)) {	
explicit_rdpem_flag [x0][y0][cIdx]	ae(v)
if(explicit_rdpem_flag[x0][y0][cIdx])	
explicit_rdpem_dir_flag [x0][y0][cIdx]	ae(v)
}	
last_sig_coeff_x_prefix	ae(v)
last_sig_coeff_y_prefix	ae(v)
if(last_sig_coeff_x_prefix > 3)	
last_sig_coeff_x_suffix	ae(v)
if(last_sig_coeff_y_prefix > 3)	
last_sig_coeff_y_suffix	ae(v)
lastScanPos = 16	
lastSubBlock = (1 << (log2TrafoSize - 2)) * (1 << (log2TrafoSize - 2)) - 1	
do {	
if(lastScanPos == 0) {	
lastScanPos = 16	
lastSubBlock--	
}	
lastScanPos--	
xS = ScanOrder[log2TrafoSize - 2][scanIdx][lastSubBlock][0]	
yS = ScanOrder[log2TrafoSize - 2][scanIdx][lastSubBlock][1]	
xC = (xS << 2) + ScanOrder[2][scanIdx][lastScanPos][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][lastScanPos][1]	
} while((xC != LastSignificantCoeffX) (yC != LastSignificantCoeffY))	
for(i = lastSubBlock; i >= 0; i--) {	
xS = ScanOrder[log2TrafoSize - 2][scanIdx][i][0]	
yS = ScanOrder[log2TrafoSize - 2][scanIdx][i][1]	
escapeDataPresent = 0	
inferSbDcSigCoeffFlag = 0	
if((i < lastSubBlock) && (i > 0)) {	
coded_sub_block_flag [xS][yS]	ae(v)
inferSbDcSigCoeffFlag = 1	
}	
for(n = (i == lastSubBlock) ? lastScanPos - 1 : 15; n >= 0; n--) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(coded_sub_block_flag[xS][yS] && (n > 0 !inferSbDcSigCoeffFlag)) {	
sig_coeff_flag [xC][yC]	ae(v)
if(sig_coeff_flag[xC][yC])	
inferSbDcSigCoeffFlag = 0	
}	
}	

firstSigScanPos = 16	
lastSigScanPos = -1	
numGreater1Flag = 0	
lastGreater1ScanPos = -1	
for(n = 15; n >= 0; n--) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(sig_coeff_flag[xC][yC]) {	
if(numGreater1Flag < 8) {	
coeff_abs_level_greater1_flag[n]	ae(v)
numGreater1Flag++	
if(coeff_abs_level_greater1_flag[n] && lastGreater1ScanPos == -1)	
lastGreater1ScanPos = n	
else if(coeff_abs_level_greater1_flag[n])	
escapeDataPresent = 1	
} else	
escapeDataPresent = 1	
if(lastSigScanPos == -1)	
lastSigScanPos = n	
firstSigScanPos = n	
}	
}	
if(cu_transquant_bypass_flag	
(CuPredMode[x0][y0] == MODE_INTRA &&	
implicit_rdpem_enabled_flag && transform_skip_flag[x0][y0][cIdx] &&	
(predModeIntra == 10 predModeIntra == 26))	
explicit_rdpem_flag[x0][y0][cIdx])	
signHidden = 0	
else	
signHidden = lastSigScanPos - firstSigScanPos > 3	
if(lastGreater1ScanPos != -1) {	
coeff_abs_level_greater2_flag[lastGreater1ScanPos]	ae(v)
if(coeff_abs_level_greater2_flag[lastGreater1ScanPos])	
escapeDataPresent = 1	
}	
for(n = 15; n >= 0; n--) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(sig_coeff_flag[xC][yC] &&	
(!sign_data_hiding_enabled_flag !signHidden (n != firstSigScanPos)))	
coeff_sign_flag[n]	ae(v)
}	
numSigCoeff = 0	
sumAbsLevel = 0	
for(n = 15; n >= 0; n--) {	
xC = (xS << 2) + ScanOrder[2][scanIdx][n][0]	
yC = (yS << 2) + ScanOrder[2][scanIdx][n][1]	
if(sig_coeff_flag[xC][yC]) {	

baseLevel = 1 + coeff_abs_level_greater1_flag[n] + coeff_abs_level_greater2_flag[n]	
if(baseLevel == ((numSigCoeff < 8) ? (n == lastGreater1ScanPos) ? 3 : 2) : 1))	
coeff_abs_level_remaining[n]	ae(v)
TransCoeffLevel[x0][y0][cldx][xC][yC] = (coeff_abs_level_remaining[n] + baseLevel) * (1 - 2 * coeff_sign_flag[n])	
if(sign_data_hiding_enabled_flag && signHidden) {	
sumAbsLevel += (coeff_abs_level_remaining[n] + baseLevel)	
if((n == firstSigScanPos) && ((sumAbsLevel % 2) == 1))	
TransCoeffLevel[x0][y0][cldx][xC][yC] = -TransCoeffLevel[x0][y0][cldx][xC][yC]	
}	
numSigCoeff++	
}	
}	
}	
}	

7.3.8.12 Cross-component prediction syntax

cross_comp_pred(x0, y0, c) {	Descriptor
log2_res_scale_abs_plus1[c]	ae(v)
if(log2_res_scale_abs_plus1[c] != 0)	
res_scale_sign_flag[c]	ae(v)
}	

7.3.8.13 Palette syntax

palette_coding(x0, y0, nCbS) {	Descriptor
palettePredictionFinished = 0	
NumPredictedPaletteEntries = 0	
for(predictorEntryIdx = 0; predictorEntryIdx < PredictorPaletteSize && !palettePredictionFinished && NumPredictedPaletteEntries < palette_max_size; predictorEntryIdx++) {	
palette_predictor_run	ae(v)
if(palette_predictor_run != 1) {	
if(palette_predictor_run > 1)	
predictorEntryIdx += palette_predictor_run - 1	
PalettePredictorEntryReuseFlags[predictorEntryIdx] = 1	
NumPredictedPaletteEntries++	
} else	
palettePredictionFinished = 1	
}	
if(NumPredictedPaletteEntries < palette_max_size)	
num_signalled_palette_entries	ae(v)
numComps = (ChromaArrayType == 0) ? 1 : 3	

for(cIdx = 0; cIdx < numComps; cIdx++)	
for(i = 0; i < num_signalled_palette_entries; i++)	
new_palette_entries [cIdx][i]	ae(v)
if(CurrentPaletteSize != 0)	
palette_escape_val_present_flag	ae(v)
if(MaxPaletteIndex > 0) {	
num_palette_indices_minus1	ae(v)
adjust = 0	
for(i = 0; i <= num_palette_indices_minus1; i++) {	
if(MaxPaletteIndex - adjust > 0) {	
palette_idx_idc	ae(v)
PaletteIndexIdc[i] = palette_idx_idc	
}	
adjust = 1	
}	
copy_above_indices_for_final_run_flag	ae(v)
palette_transpose_flag	ae(v)
}	
if(palette_escape_val_present_flag) {	
delta_qp()	
if(!cu_transquant_bypass_flag)	
chroma_qp_offset()	
}	
remainingNumIndices = num_palette_indices_minus1 + 1	
PaletteScanPos = 0	
log2BlockSize = Log2(nCbS)	
while(PaletteScanPos < nCbS * nCbS) {	
xC = x0 + ScanOrder[log2BlockSize][3][PaletteScanPos][0]	
yC = y0 + ScanOrder[log2BlockSize][3][PaletteScanPos][1]	
if(PaletteScanPos > 0) {	
xcPrev = x0 + ScanOrder[log2BlockSize][3][PaletteScanPos - 1][0]	
ycPrev = y0 + ScanOrder[log2BlockSize][3][PaletteScanPos - 1][1]	
}	
PaletteRunMinus1 = nCbS * nCbS - PaletteScanPos - 1	
RunToEnd = 1	
CopyAboveIndicesFlag[xC][yC] = 0	
if(MaxPaletteIndex > 0)	
if(PaletteScanPos >= nCbS && CopyAboveIndicesFlag[xcPrev][ycPrev] == 0)	
if(remainingNumIndices > 0 && PaletteScanPos < nCbS * nCbS - 1) {	
copy_above_palette_indices_flag	ae(v)
CopyAboveIndicesFlag[xC][yC] = copy_above_palette_indices_flag	
} else	
if(PaletteScanPos == nCbS * nCbS - 1 && remainingNumIndices > 0)	
CopyAboveIndicesFlag[xC][yC] = 0	
else	
CopyAboveIndicesFlag[xC][yC] = 1	

if(CopyAboveIndicesFlag[xC][yC] == 0) {	
currNumIndices = num_palette_indices_minus1 + 1 – remainingNumIndices	
CurrPaletteIndex = PaletteIndexIdx[currNumIndices]	
}	
if(MaxPaletteIndex > 0) {	
if(CopyAboveIndicesFlag[xC][yC] == 0)	
remainingNumIndices – = 1	
if(remainingNumIndices > 0 CopyAboveIndicesFlag[xC][yC] != copy_above_indices_for_final_run_flag) {	
PaletteMaxRunMinus1 = nCbS * nCbS – PaletteScanPos – 1 – remainingNumIndices – copy_above_indices_for_final_run_flag	
RunToEnd = 0	
if(PaletteMaxRunMinus1 > 0) {	
palette_run_prefix	ae(v)
if((palette_run_prefix > 1) && (PaletteMaxRunMinus1 != (1 << (palette_run_prefix – 1))))	
palette_run_suffix	ae(v)
}	
}	
}	
runPos = 0	
while (runPos <= PaletteRunMinus1) {	
xR = x0 + ScanOrder[log2BlockSize][3][PaletteScanPos][0]	
yR = y0 + ScanOrder[log2BlockSize][3][PaletteScanPos][1]	
if(CopyAboveIndicesFlag[xC][yC] == 0) {	
CopyAboveIndicesFlag[xR][yR] = 0	
PaletteIndexMap[xR][yR] = CurrPaletteIndex	
} else {	
CopyAboveIndicesFlag[xR][yR] = 1	
PaletteIndexMap[xR][yR] = PaletteIndexMap[xR][yR – 1]	
}	
runPos++	
PaletteScanPos++	
}	
}	
if(palette_escape_val_present_flag) {	
for(cIdx = 0; cIdx < numComps; cIdx++)	
for(sPos = 0; sPos < nCbS * nCbS; sPos++) {	
xC = x0 + ScanOrder[log2BlockSize][3][sPos][0]	
yC = y0 + ScanOrder[log2BlockSize][3][sPos][1]	
if(PaletteIndexMap[xC][yC] == MaxPaletteIndex)	
if(cIdx == 0 (xC % 2 == 0 && yC % 2 == 0 && ChromaArrayType == 1) (xC % 2 == 0 && !palette_transpose_flag && ChromaArrayType == 2) (yC % 2 == 0 && palette_transpose_flag && ChromaArrayType == 2) ChromaArrayType == 3) {	

palette_escape_val	ae(v)
PaletteEscapeVal[cIdx][xC][yC] = palette_escape_val	
}	
}	
}	
}	

7.3.8.14 Delta QP syntax

delta_qp() {	Descriptor
if(cu_qp_delta_enabled_flag && !IsCuQpDeltaCoded) {	
IsCuQpDeltaCoded = 1	
cu_qp_delta_abs	ae(v)
if(cu_qp_delta_abs) {	
cu_qp_delta_sign_flag	ae(v)
CuQpDeltaVal = cu_qp_delta_abs * (1 - 2 * cu_qp_delta_sign_flag)	
}	
}	
}	

7.3.8.15 Chroma QP offset syntax

chroma_qp_offset() {	Descriptor
if(cu_chroma_qp_offset_enabled_flag && !IsCuChromaQpOffsetCoded) {	
cu_chroma_qp_offset_flag	ae(v)
if(cu_chroma_qp_offset_flag && chroma_qp_offset_list_len_minus1 > 0)	
cu_chroma_qp_offset_idx	ae(v)
}	
}	

7.4 Semantics

7.4.1 General

Semantics associated with the syntax structures and with the syntax elements within these structures are specified in this clause. When the semantics of a syntax element are specified using a table or a set of tables, any values that are not specified in the table(s) shall not be present in the bitstream unless otherwise specified in this Specification.

7.4.2 NAL unit semantics

7.4.2.1 General NAL unit semantics

NumBytesInNalUnit specifies the size of the NAL unit in bytes. This value is required for decoding of the NAL unit. Some form of demarcation of NAL unit boundaries is necessary to enable inference of NumBytesInNalUnit. One such demarcation method is specified in Annex B for the byte stream format. Other methods of demarcation may be specified outside of this Specification.

NOTE 1 – The video coding layer (VCL) is specified to efficiently represent the content of the video data. The NAL is specified to format that data and provide header information in a manner appropriate for conveyance on a variety of communication channels or storage media. All data are contained in NAL units, each of which contains an integer number of bytes. A NAL unit specifies a generic format for use in both packet-oriented and bitstream systems. The format of NAL units for both packet-oriented transport and byte stream is identical except that each NAL unit can be preceded by a start code prefix and extra padding bytes in the byte stream format specified in Annex B.

rbsp_byte[i] is the *i*-th byte of an RBSP. An RBSP is specified as an ordered sequence of bytes as follows:

The RBSP contains a string of data bits (SODB) as follows:

- If the SODB is empty (i.e., zero bits in length), the RBSP is also empty.
- Otherwise, the RBSP contains the SODB as follows:
 - 1) The first byte of the RBSP contains the first (most significant, left-most) eight bits of the SODB; the next byte of the RBSP contains the next eight bits of the SODB, etc., until fewer than eight bits of the SODB remain.
 - 2) The **rbsp_trailing_bits()** syntax structure is present after the SODB as follows:
 - i) The first (most significant, left-most) bits of the final RBSP byte contain the remaining bits of the SODB (if any).
 - ii) The next bit consists of a single bit equal to 1 (i.e., **rbsp_stop_one_bit**).
 - iii) When the **rbsp_stop_one_bit** is not the last bit of a byte-aligned byte, one or more zero-valued bits (i.e., instances of **rbsp_alignment_zero_bit**) are present to result in byte alignment.
 - 3) One or more **cabac_zero_word** 16-bit syntax elements equal to 0x0000 may be present in some RBSPs after the **rbsp_trailing_bits()** at the end of the RBSP.

Syntax structures having these RBSP properties are denoted in the syntax tables using an "**_rbsp**" suffix. These structures are carried within NAL units as the content of the **rbsp_byte[i]** data bytes. The association of the RBSP syntax structures to the NAL units is as specified in Table 7-1.

NOTE 2 – When the boundaries of the RBSP are known, the decoder can extract the SODB from the RBSP by concatenating the bits of the bytes of the RBSP and discarding the **rbsp_stop_one_bit**, which is the last (least significant, right-most) bit equal to 1, and discarding any following (less significant, farther to the right) bits that follow it, which are equal to 0. The data necessary for the decoding process is contained in the SODB part of the RBSP.

emulation_prevention_three_byte is a byte equal to 0x03. When an **emulation_prevention_three_byte** is present in the NAL unit, it shall be discarded by the decoding process.

The last byte of the NAL unit shall not be equal to 0x00.

Within the NAL unit, the following three-byte sequences shall not occur at any byte-aligned position:

- 0x000000
- 0x000001
- 0x000002

Within the NAL unit, any four-byte sequence that starts with 0x000003 other than the following sequences shall not occur at any byte-aligned position:

- 0x00000300
- 0x00000301
- 0x00000302
- 0x00000303

7.4.2.2 NAL unit header semantics

forbidden_zero_bit shall be equal to 0.

nal_unit_type specifies the type of RBSP data structure contained in the NAL unit as specified in Table 7-1.

NAL units that have **nal_unit_type** in the range of UNSPEC48..UNSPEC63, inclusive, for which semantics are not specified, shall not affect the decoding process specified in this Specification.

NOTE 1 – NAL unit types in the range of UNSPEC48..UNSPEC63 may be used as determined by the application. No decoding process for these values of **nal_unit_type** is specified in this Specification. Since different applications might use these NAL unit types for different purposes, particular care must be exercised in the design of encoders that generate NAL units with these **nal_unit_type** values, and in the design of decoders that interpret the content of NAL units with these **nal_unit_type** values. This Specification does not define any management for these values. These **nal_unit_type** values might only be suitable for use in contexts in which "collisions" of usage (i.e., different definitions of the meaning of the NAL unit content for the same **nal_unit_type** value) are unimportant, or not possible, or are managed – e.g., defined or managed in the controlling application or transport specification, or by controlling the environment in which bitstreams are distributed.

For purposes other than determining the amount of data in the decoding units of the bitstream (as specified in Annex C), decoders shall ignore (remove from the bitstream and discard) the contents of all NAL units that use reserved values of **nal_unit_type**.

NOTE 2 – This requirement allows future definition of compatible extensions to this Specification.

Table 7-1 – NAL unit type codes and NAL unit type classes

nal_unit_type	Name of nal_unit_type	Content of NAL unit and RBSP syntax structure	NAL unit type class
0 1	TRAIL_N TRAIL_R	Coded slice segment of a non-TSA, non-STSA trailing picture slice_segment_layer_rbsp()	VCL
2 3	TSA_N TSA_R	Coded slice segment of a TSA picture slice_segment_layer_rbsp()	VCL
4 5	STSA_N STSA_R	Coded slice segment of an STSA picture slice_segment_layer_rbsp()	VCL
6 7	RADL_N RADL_R	Coded slice segment of a RADL picture slice_segment_layer_rbsp()	VCL
8 9	RASL_N RASL_R	Coded slice segment of a RASL picture slice_segment_layer_rbsp()	VCL
10 12 14	RSV_VCL_N10 RSV_VCL_N12 RSV_VCL_N14	Reserved non-IRAP SLNR VCL NAL unit types	VCL
11 13 15	RSV_VCL_R11 RSV_VCL_R13 RSV_VCL_R15	Reserved non-IRAP sub-layer reference VCL NAL unit types	VCL
16 17 18	BLA_W_LP BLA_W_RADL BLA_N_LP	Coded slice segment of a BLA picture slice_segment_layer_rbsp()	VCL
19 20	IDR_W_RADL IDR_N_LP	Coded slice segment of an IDR picture slice_segment_layer_rbsp()	VCL
21	CRA_NUT	Coded slice segment of a CRA picture slice_segment_layer_rbsp()	VCL
22 23	RSV_IRAP_VCL22 RSV_IRAP_VCL23	Reserved IRAP VCL NAL unit types	VCL
24..31	RSV_VCL24.. RSV_VCL31	Reserved non-IRAP VCL NAL unit types	VCL
32	VPS_NUT	Video parameter set video_parameter_set_rbsp()	non-VCL
33	SPS_NUT	Sequence parameter set seq_parameter_set_rbsp()	non-VCL
34	PPS_NUT	Picture parameter set pic_parameter_set_rbsp()	non-VCL
35	AUD_NUT	Access unit delimiter access_unit_delimiter_rbsp()	non-VCL
36	EOS_NUT	End of sequence end_of_seq_rbsp()	non-VCL
37	EOB_NUT	End of bitstream end_of_bitstream_rbsp()	non-VCL
38	FD_NUT	Filler data filler_data_rbsp()	non-VCL
39 40	PREFIX_SEI_NUT SUFFIX_SEI_NUT	Supplemental enhancement information sei_rbsp()	non-VCL
41..47	RSV_NVCL41.. RSV_NVCL47	Reserved	non-VCL
48..63	UNSPEC48.. UNSPEC63	Unspecified	non-VCL

NOTE 3 – A clean random access (CRA) picture may have associated random access skipped leading (RASL) or random access decodable leading (RADL) pictures present in the bitstream.

NOTE 4 – A broken link access (BLA) picture having `nal_unit_type` equal to `BLA_W_LP` may have associated RASL or RADL pictures present in the bitstream. A BLA picture having `nal_unit_type` equal to `BLA_W_RADL` does not have associated RASL pictures present in the bitstream, but may have associated RADL pictures in the bitstream. A BLA picture having `nal_unit_type` equal to `BLA_N_LP` does not have associated leading pictures present in the bitstream.

NOTE 5 – An instantaneous decoding refresh (IDR) picture having `nal_unit_type` equal to `IDR_N_LP` does not have associated leading pictures present in the bitstream. An IDR picture having `nal_unit_type` equal to `IDR_W_RADL` does not have associated RASL pictures present in the bitstream, but may have associated RADL pictures in the bitstream.

NOTE 6 – A sub-layer non-reference (SLNR) picture is not included in any of `RefPicSetStCurrBefore`, `RefPicSetStCurrAfter` and `RefPicSetLtCurr` of any picture with the same value of `TemporalId`, and may be discarded without affecting the decodability of other pictures with the same value of `TemporalId`.

All coded slice segment NAL units of an access unit shall have the same value of `nal_unit_type`. A picture or an access unit is also referred to as having a `nal_unit_type` equal to the `nal_unit_type` of the coded slice segment NAL units of the picture or the access unit.

If a picture has `nal_unit_type` equal to `TRAIL_N`, `TSA_N`, `STSA_N`, `RADL_N`, `RASL_N`, `RSV_VCL_N10`, `RSV_VCL_N12` or `RSV_VCL_N14`, the picture is an SLNR picture. Otherwise, the picture is a sub-layer reference picture.

Each picture, other than the first picture in the bitstream in decoding order, is considered to be associated with the previous intra random access point (IRAP) picture in decoding order.

When a picture is a leading picture, it shall be a RADL or RASL picture.

When a picture is a trailing picture, it shall not be a RADL or RASL picture.

When a picture is a leading picture, it shall precede, in decoding order, all trailing pictures that are associated with the same IRAP picture.

No RASL pictures shall be present in the bitstream that are associated with a BLA picture having `nal_unit_type` equal to `BLA_W_RADL` or `BLA_N_LP`.

No RASL pictures shall be present in the bitstream that are associated with an IDR picture.

No RADL pictures shall be present in the bitstream that are associated with a BLA picture having `nal_unit_type` equal to `BLA_N_LP` or that are associated with an IDR picture having `nal_unit_type` equal to `IDR_N_LP`.

NOTE 7 – It is possible to perform random access at the position of an IRAP access unit by discarding all access units before the IRAP access unit (and to correctly decode the IRAP picture and all the subsequent non-RASL pictures in decoding order), provided each parameter set is available (either in the bitstream or by external means not specified in this Specification) when it needs to be activated.

Any picture that has `PicOutputFlag` equal to 1 that precedes an IRAP picture in decoding order shall precede the IRAP picture in output order and shall precede any RADL picture associated with the IRAP picture in output order.

Any RASL picture associated with a CRA or BLA picture shall precede any RADL picture associated with the CRA or BLA picture in output order.

Any RASL picture associated with a CRA picture shall follow, in output order, any IRAP picture that precedes the CRA picture in decoding order.

When `sps_temporal_id_nesting_flag` is equal to 1 and `TemporalId` is greater than 0, the `nal_unit_type` shall be equal to `TSA_R`, `TSA_N`, `RADL_R`, `RADL_N`, `RASL_R` or `RASL_N`.

`nuh_layer_id` specifies the identifier of the layer to which a VCL NAL unit belongs or the identifier of a layer to which a non-VCL NAL unit applies. The value of `nuh_layer_id` shall be in the range of 0 to 62, inclusive. The value of 63 may be specified in the future by ITU-T | ISO/IEC. For purposes other than determining the amount of data in the decoding units of the bitstream, decoders shall ignore all data that follow the value 63 for `nuh_layer_id` in a NAL unit, and decoders conforming to a profile specified in Annex A and not supporting the independent non-base layer decoding (INBLD) capability specified in Annex F shall ignore (i.e., remove from the bitstream and discard) all NAL units with values of `nuh_layer_id` not equal to 0.

NOTE 8 – The value of 63 for `nuh_layer_id` may be used to indicate an extended layer identifier in a future extension of this Specification.

The value of `nuh_layer_id` shall be the same for all VCL NAL units of a coded picture. The value of `nuh_layer_id` of a coded picture is the value of the `nuh_layer_id` of the VCL NAL units of the coded picture.

When `nal_unit_type` is equal to `EOB_NUT`, the value of `nuh_layer_id` shall be equal to 0.

nuh_temporal_id_plus1 minus 1 specifies a temporal identifier for the NAL unit. The value of **nuh_temporal_id_plus1** shall not be equal to 0.

The variable **TemporalId** is specified as follows:

$$\text{TemporalId} = \text{nuh_temporal_id_plus1} - 1 \quad (7-1)$$

When **nal_unit_type** is in the range of **BLA_W_LP** to **RSV_IRAP_VCL23**, inclusive, i.e., the coded slice segment belongs to an IRAP picture, **TemporalId** shall be equal to 0.

When **nal_unit_type** is equal to **TSA_R** or **TSA_N**, **TemporalId** shall not be equal to 0.

When **nuh_layer_id** is equal to 0 and **nal_unit_type** is equal to **STSA_R** or **STSA_N**, **TemporalId** shall not be equal to 0.

The value of **TemporalId** shall be the same for all VCL NAL units of an access unit. The value of **TemporalId** of a coded picture or an access unit is the value of the **TemporalId** of the VCL NAL units of the coded picture or the access unit. The value of **TemporalId** of a sub-layer representation is the greatest value of **TemporalId** of all VCL NAL units in the sub-layer representation.

The value of **TemporalId** for non-VCL NAL units is constrained as follows:

- If **nal_unit_type** is equal to **VPS_NUT** or **SPS_NUT**, **TemporalId** shall be equal to 0 and the **TemporalId** of the access unit containing the NAL unit shall be equal to 0.
- Otherwise if **nal_unit_type** is equal to **EOS_NUT** or **EOB_NUT**, **TemporalId** shall be equal to 0.
- Otherwise, if **nal_unit_type** is equal to **AUD_NUT** or **FD_NUT**, **TemporalId** shall be equal to the **TemporalId** of the access unit containing the NAL unit.
- Otherwise, **TemporalId** shall be greater than or equal to the **TemporalId** of the access unit containing the NAL unit.

NOTE 9 – When the NAL unit is a non-VCL NAL unit, the value of **TemporalId** is equal to the minimum value of the **TemporalId** values of all access units to which the non-VCL NAL unit applies. When **nal_unit_type** is equal to **PPS_NUT**, **TemporalId** may be greater than or equal to the **TemporalId** of the containing access unit, as all picture parameter sets (PPSs) may be included in the beginning of a bitstream, wherein the first coded picture has **TemporalId** equal to 0. When **nal_unit_type** is equal to **PREFIX_SEI_NUT** or **SUFFIX_SEI_NUT**, **TemporalId** may be greater than or equal to the **TemporalId** of the containing access unit, as an SEI NAL unit may contain information, e.g., in a buffering period SEI message or a picture timing SEI message, that applies to a bitstream subset that includes access units for which the **TemporalId** values are greater than the **TemporalId** of the access unit containing the SEI NAL unit.

7.4.2.3 Encapsulation of an SODB within an RBSP (informative)

This clause does not form an integral part of this Specification.

The form of encapsulation of an SODB within an RBSP and the use of the **emulation_prevention_three_byte** for encapsulation of an RBSP within a NAL unit is described for the following purposes:

- To prevent the emulation of start codes within NAL units while allowing any arbitrary SODB to be represented within a NAL unit,
- To enable identification of the end of the SODB within the NAL unit by searching the RBSP for the **rbsp_stop_one_bit** starting at the end of the RBSP,
- To enable a NAL unit to have a size greater than that of the SODB under some circumstances (using one or more **cabac_zero_word** syntax elements).

The encoder can produce a NAL unit from an RBSP by the following procedure:

1. The RBSP data are searched for byte-aligned bits of the following binary patterns:

'00000000 00000000 000000xx' (where 'xx' represents any two-bit pattern: '00', '01', '10', or '11'),

and a byte equal to 0x03 is inserted to replace the bit pattern with the pattern:

'00000000 00000000 00000011 000000xx',

and finally, when the last byte of the RBSP data is equal to 0x00 (which can only occur when the RBSP ends in a **cabac_zero_word**), a final byte equal to 0x03 is appended to the end of the data. The last zero byte of a byte-aligned three-byte sequence 0x000000 in the RBSP (which is replaced by the four-byte sequence 0x00000300) is taken into account when searching the RBSP data for the next occurrence of byte-aligned bits with the binary patterns specified above.

2. The resulting sequence of bytes is then prefixed with the NAL unit header, within which the **nal_unit_type** indicates the type of RBSP data structure in the NAL unit.

The process specified above results in the construction of the entire NAL unit.

This process can allow any SODB to be represented in a NAL unit while ensuring both of the following:

- No byte-aligned start code prefix is emulated within the NAL unit.
- No sequence of 8 zero-valued bits followed by a start code prefix, regardless of byte-alignment, is emulated within the NAL unit.

7.4.2.4 Order of NAL units and association to coded pictures, access units and coded video sequences

7.4.2.4.1 General

This clause specifies constraints on the order of NAL units in the bitstream.

Any order of NAL units in the bitstream obeying these constraints is referred to in the text as the decoding order of NAL units. Within a NAL unit, the syntax in clauses 7.3, D.2 and E.2 specifies the decoding order of syntax elements. Decoders shall be capable of receiving NAL units and their syntax elements in decoding order.

7.4.2.4.2 Order of VPS, SPS and PPS RBSPs and their activation

This clause specifies the activation process of video parameter sets (VPSs), sequence parameter sets (SPSs) and PPSs.

NOTE 1 – The VPS, SPS and PPS mechanism decouples the transmission of infrequently changing information from the transmission of coded block data. VPSs, SPSs and PPSs may, in some applications, be conveyed "out-of-band".

A PPS RBSP includes parameters that can be referred to by the coded slice segment NAL units of one or more coded pictures. Each PPS RBSP is initially considered not active for the base layer at the start of the operation of the decoding process. At most one PPS RBSP is considered active for the base layer at any given moment during the operation of the decoding process, and the activation of any particular PPS RBSP for the base layer results in the deactivation of the previously-active PPS RBSP for the base layer (if any).

When a PPS RBSP (with a particular value of `pps_pic_parameter_set_id`) is not active for the base layer and it is referred to by a coded slice segment NAL unit with `nuh_layer_id` equal to 0 (using a value of `slice_pic_parameter_set_id` equal to the `pps_pic_parameter_set_id` value), it is activated for the base layer. This PPS RBSP is called the active PPS RBSP for the base layer until it is deactivated by the activation of another PPS RBSP for the base layer. A PPS RBSP, with that particular value of `pps_pic_parameter_set_id`, shall be available to the decoding process prior to its activation, included in at least one access unit with `TemporalId` less than or equal to the `TemporalId` of the PPS NAL unit or provided through external means, and the PPS NAL unit containing the PPS RBSP shall have `nuh_layer_id` equal to 0.

Any PPS NAL unit containing the value of `pps_pic_parameter_set_id` for the active PPS RBSP for a coded picture (and consequently for the layer containing the coded picture) shall have the same content as that of the active PPS RBSP for the coded picture, unless it follows the last VCL NAL unit of the coded picture and precedes the first VCL NAL unit of another coded picture.

An SPS RBSP includes parameters that can be referred to by one or more PPS RBSPs or one or more SEI NAL units containing an active parameter sets SEI message. Each SPS RBSP is initially considered not active for the base layer at the start of the operation of the decoding process. At most one SPS RBSP is considered active for the base layer at any given moment during the operation of the decoding process, and the activation of any particular SPS RBSP for the base layer results in the deactivation of the previously-active SPS RBSP for the base layer (if any).

When an SPS RBSP (with a particular value of `sps_seq_parameter_set_id`) is not already active for the base layer and it is referred to by activation of a PPS RBSP (in which `pps_seq_parameter_set_id` is equal to the `sps_seq_parameter_set_id` value) for the base layer or, when `vps_base_layer_internal_flag` is equal to 1 and `vps_base_layer_available_flag` is equal to 1, is referred to by an SEI NAL unit containing an active parameter sets SEI message (in which `active_seq_parameter_set_id[0]` is equal to the `sps_seq_parameter_set_id` value), it is activated for the base layer. This SPS RBSP is called the active SPS RBSP for the base layer until it is deactivated by the activation of another SPS RBSP for the base layer. An SPS RBSP, with that particular value of `sps_seq_parameter_set_id`, shall be available to the decoding process prior to its activation, included in at least one access unit with `TemporalId` equal to 0 or provided through external means, and the SPS NAL unit containing the SPS RBSP shall have `nuh_layer_id` equal to 0. An activated SPS RBSP for the base layer shall remain active for the entire coded video sequence (CVS).

NOTE 2 – Because an IRAP access unit with `NoRasOutputFlag` equal to 1 begins a new CVS and an activated SPS RBSP for the base layer must remain active for the entire CVS, an SPS RBSP can only be activated for the base layer by an active parameter sets SEI message when the active parameter sets SEI message is part of an IRAP access unit with `NoRasOutputFlag` equal to 1.

Any SPS NAL unit with `nuh_layer_id` equal to 0 containing the value of `sps_seq_parameter_set_id` for the active SPS RBSP for the base layer for a CVS shall have the same content as that of the active SPS RBSP for the base layer for the CVS, unless it follows the last access unit of the CVS and precedes the first VCL NAL unit and the first SEI NAL unit containing an active parameter sets SEI message (when present) of another CVS.

A VPS RBSP includes parameters that can be referred to by one or more SPS RBSPs or one or more SEI NAL units containing an active parameter sets SEI message. Each VPS RBSP is initially considered not active at the start of the operation of the decoding process. At most one VPS RBSP is considered active at any given moment during the operation of the decoding process, and the activation of any particular VPS RBSP results in the deactivation of the previously-active VPS RBSP (if any).

When a VPS RBSP (with a particular value of `vps_video_parameter_set_id`) is not already active and it is referred to by activation of an SPS RBSP (in which `sps_video_parameter_set_id` is equal to the `vps_video_parameter_set_id` value) for the base layer, or is referred to by an SEI NAL unit containing an active parameter sets SEI message (in which `active_video_parameter_set_id` is equal to the `vps_video_parameter_set_id` value), it is activated. This VPS RBSP is called the active VPS RBSP until it is deactivated by the activation of another VPS RBSP. A VPS RBSP, with that particular value of `vps_video_parameter_set_id`, shall be available to the decoding process prior to its activation, included in at least one access unit with `TemporalId` equal to 0 or provided through external means, and the VPS NAL unit containing the VPS RBSP shall have `nuh_layer_id` equal to 0. An activated VPS RBSP shall remain active for the entire CVS.

NOTE 3 – Because an IRAP access unit with `NoRasOutputFlag` equal to 1 begins a new CVS and an activated VPS RBSP must remain active for the entire CVS, a VPS RBSP can only be activated by an active parameter sets SEI message when the active parameter sets SEI message is part of an IRAP access unit with `NoRasOutputFlag` equal to 1.

Any VPS NAL unit containing the value of `vps_video_parameter_set_id` for the active VPS RBSP for a CVS shall have the same content as that of the active VPS RBSP for the CVS, unless it follows the last access unit of the CVS and precedes the first VCL NAL unit, the first SPS NAL unit and the first SEI NAL unit containing an active parameter sets SEI message (when present) of another CVS.

NOTE 4 – If VPS RBSP, SPS RBSP or PPS RBSP are conveyed within the bitstream, these constraints impose an order constraint on the NAL units that contain the VPS RBSP, SPS RBSP or PPS RBSP, respectively. Otherwise (VPS RBSP, SPS RBSP or PPS RBSP are conveyed by other means not specified in this Specification), they must be available to the decoding process in a timely fashion such that these constraints are obeyed.

All constraints that are expressed on the relationship between the values of the syntax elements and the values of variables derived from those syntax elements in VPSs, SPSs and PPSs and other syntax elements are expressions of constraints that apply only to the active VPS RBSP, the active SPS RBSP for the base layer and the active PPS RBSP for the base layer. If any VPS RBSP, SPS RBSP and PPS RBSP is present that is never activated in the bitstream, its syntax elements shall have values that would conform to the specified constraints if it was activated by reference in an otherwise conforming bitstream.

NOTE 5 – In the context of this clause, activation of a parameter set RBSP is for the base layer only. Thus, the constraint above on never-activated parameter set RBSPs applies to those parameter set RBSPs with `nuh_layer_id` equal to 0 only, because parameter set RBSPs with `nuh_layer_id` greater than 0 are not allowed to be referred to by the base layer.

During operation of the decoding process (see clause 8), the values of parameters of the active VPS RBSP, the active SPS RBSP for the base layer and the active PPS RBSP for the base layer are considered in effect. For interpretation of SEI messages, the values of the active VPS RBSP, the active SPS RBSP for the base layer and the active PPS RBSP for the base layer for the operation of the decoding process for the VCL NAL units of the coded picture with `nuh_layer_id` equal to 0 in the same access unit are considered in effect unless otherwise specified in the SEI message semantics.

7.4.2.4.3 Order of access units and association to CVSs

A bitstream conforming to this Specification consists of one or more CVSs.

A CVS consists of one or more access units. The order of NAL units and coded pictures and their association to access units is described in clause 7.4.2.4.4.

The first access unit of a CVS is an IRAP access unit with `NoRasOutputFlag` equal to 1.

It is a requirement of bitstream conformance that, when present, the next access unit after an access unit that contains an end of sequence NAL unit shall be an IRAP access unit, which may be an IDR access unit, a BLA access unit, or a CRA access unit.

7.4.2.4.4 Order of NAL units and coded pictures and their association to access units

This clause specifies the order of NAL units and coded pictures and their association to access units for CVSs that conform to one or more of the profiles specified in Annex A and that are decoded using the decoding process specified in clauses 2 through 10.

An access unit consists of one coded picture with `nuh_layer_id` equal to 0, zero or more VCL NAL units with `nuh_layer_id` greater than 0 and zero or more non-VCL NAL units. The association of VCL NAL units to coded pictures is described in clause 7.4.2.4.5.

The first access unit in the bitstream starts with the first NAL unit of the bitstream.

Let firstBIPicNalUnit be the first VCL NAL unit of a coded picture with nuh_layer_id equal to 0. The first of any of the following NAL units preceding firstBIPicNalUnit and succeeding the last VCL NAL unit preceding firstBIPicNalUnit, if any, specifies the start of a new access unit:

NOTE 1 – The last VCL NAL unit preceding firstBIPicNalUnit in decoding order may have nuh_layer_id greater than 0.

- access unit delimiter NAL unit with nuh_layer_id equal to 0 (when present),
- VPS NAL unit with nuh_layer_id equal to 0 (when present),
- SPS NAL unit with nuh_layer_id equal to 0 (when present),
- PPS NAL unit with nuh_layer_id equal to 0 (when present),
- Prefix SEI NAL unit with nuh_layer_id equal to 0 (when present),
- NAL units with nal_unit_type in the range of RSV_NVCL41..RSV_NVCL44 with nuh_layer_id equal to 0 (when present),
- NAL units with nal_unit_type in the range of UNSPEC48..UNSPEC55 with nuh_layer_id equal to 0 (when present).

NOTE 2 – The first NAL unit preceding firstBIPicNalUnit and succeeding the last VCL NAL unit preceding firstBIPicNalUnit, if any, can only be one of the above-listed NAL units.

When there is none of the above NAL units preceding firstBIPicNalUnit and succeeding the last VCL NAL preceding firstBIPicNalUnit, if any, firstBIPicNalUnit starts a new access unit.

The order of the coded pictures and non-VCL NAL units within an access unit shall obey the following constraints:

- When an access unit delimiter NAL unit with nuh_layer_id equal to 0 is present, it shall be the first NAL unit. There shall be at most one access unit delimiter NAL unit with nuh_layer_id equal to 0 in any access unit.
- When any VPS NAL units, SPS NAL units, PPS NAL units, prefix SEI NAL units, NAL units with nal_unit_type in the range of RSV_NVCL41..RSV_NVCL44, or NAL units with nal_unit_type in the range of UNSPEC48..UNSPEC55 are present, they shall not follow the last VCL NAL unit of the access unit.
- NAL units having nal_unit_type equal to FD_NUT or SUFFIX_SEI_NUT or in the range of RSV_NVCL45..RSV_NVCL47 or UNSPEC56..UNSPEC63 shall not precede the first VCL NAL unit of the access unit.
- When an end of sequence NAL unit with nuh_layer_id equal to 0 is present, it shall be the last NAL unit among all NAL units with nuh_layer_id equal to 0 in the access unit other than an end of bitstream NAL unit (when present).
- When an end of bitstream NAL unit is present, it shall be the last NAL unit in the access unit.

NOTE 3 – Decoders conforming to profiles specified in Annex A do not use NAL units with nuh_layer_id greater than 0, e.g., access unit delimiter NAL units with nuh_layer_id greater than 0, for access unit boundary detection, except for identification of a NAL unit as a VCL or non-VCL NAL unit.

The structure of access units not containing any NAL units with nal_unit_type equal to FD_NUT, VPS_NUT, SPS_NUT, PPS_NUT, RSV_VCL_N10, RSV_VCL_R11, RSV_VCL_N12, RSV_VCL_R13, RSV_VCL_N14 or RSV_VCL_R15, RSV_IRAP_VCL22 or RSV_IRAP_VCL23, or in the range of RSV_VCL24..RSV_VCL31, RSV_NVCL41..RSV_NVCL47 or UNSPEC48..UNSPEC63 is shown in Figure 7-1.

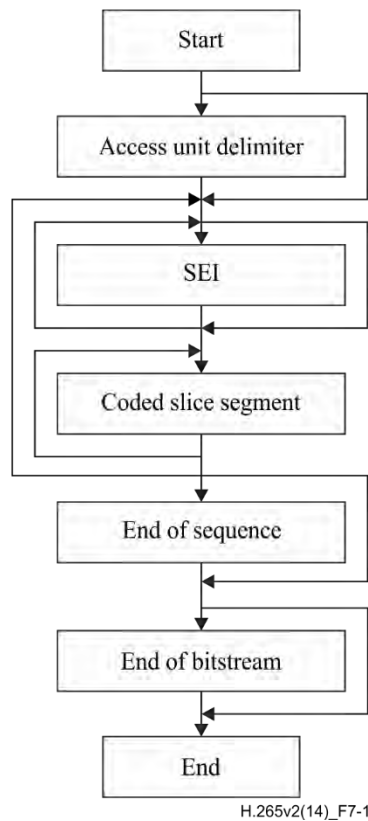


Figure 7-1 – Structure of an access unit not containing any NAL units with `nal_unit_type` equal to `FD_NUT`, `SUFFIX_SEI_NUT`, `VPS_NUT`, `SPS_NUT`, `PPS_NUT`, `RSV_VCL_N10`, `RSV_VCL_R11`, `RSV_VCL_N12`, `RSV_VCL_R13`, `RSV_VCL_N14`, `RSV_VCL_R15`, `RSV_IRAP_VCL22` or `RSV_IRAP_VCL23`, or in the range of `RSV_VCL24`..`RSV_VCL31`, `RSV_NVCL41`..`RSV_NVCL47` or `UNSPEC48`..`UNSPEC63`

7.4.2.4.5 Order of VCL NAL units and association to coded pictures

This clause specifies the order of VCL NAL units and association to coded pictures.

Each VCL NAL unit is part of a coded picture.

The order of the VCL NAL units within a coded picture is constrained as follows:

- The first VCL NAL unit of the coded picture shall have `first_slice_segment_in_pic_flag` equal to 1.
- Let `sliceSegAddrA` and `sliceSegAddrB` be the `slice_segment_address` values of any two coded slice segment NAL units A and B within the same coded picture. When either of the following conditions is true, coded slice segment NAL unit A shall precede the coded slice segment NAL unit B:
 - `TileId[CtbAddrRsToTs[sliceSegAddrA]]` is less than `TileId[CtbAddrRsToTs[sliceSegAddrB]]`.
 - `TileId[CtbAddrRsToTs[sliceSegAddrA]]` is equal to `TileId[CtbAddrRsToTs[sliceSegAddrB]]` and `CtbAddrRsToTs[sliceSegAddrA]` is less than `CtbAddrRsToTs[sliceSegAddrB]`.

7.4.3 Raw byte sequence payloads, trailing bits and byte alignment semantics

7.4.3.1 Video parameter set RBSP semantics

NOTE 1 – VPS NAL units are required to be available to the decoding process prior to their activation (either in the bitstream or by external means), as specified in clause 7.4.2.4.2. However, the VPS RBSP contains information that is not necessary for operation of the decoding process specified in clauses 2 through 10 of this Specification. For purposes other than determining the amount of data in the decoding units of the bitstream (as specified in Annex C), decoders conforming to a profile specified in Annex A but not supporting the INBLD capability specified in Annex F may ignore (remove from the bitstream and discard) the content of all VPS NAL units.

Any two instances of the syntax structure `hrd_parameters()` included in a VPS RBSP shall not have the same content.

`vps_video_parameter_set_id` identifies the VPS for reference by other syntax elements.

vps_base_layer_internal_flag and **vps_base_layer_available_flag** specify the following:

- If **vps_base_layer_internal_flag** is equal to 1 and **vps_base_layer_available_flag** is equal to 1, the base layer is present in the bitstream.
- Otherwise, if **vps_base_layer_internal_flag** is equal to 0 and **vps_base_layer_available_flag** is equal to 1, the base layer is provided by an external means not specified in this Specification.
- Otherwise, if **vps_base_layer_internal_flag** is equal to 1 and **vps_base_layer_available_flag** is equal to 0, the base layer is not available (neither present in the bitstream nor provided by external means) but the VPS includes information of the base layer as if it were present in the bitstream.
- Otherwise (**vps_base_layer_internal_flag** is equal to 0 and **vps_base_layer_available_flag** is equal to 0), the base layer is not available (neither present in the bitstream nor provided by external means) but the VPS includes information of the base layer as if it were provided by an external means not specified in this Specification.

vps_max_layers_minus1 plus 1 specifies the maximum allowed number of layers in each CVS referring to the VPS. It is a requirement of bitstream conformance that, when **vps_base_layer_internal_flag** is equal to 0, **vps_max_layers_minus1** shall be greater than 0. **vps_max_layers_minus1** shall be less than 63 in bitstreams conforming to this version of this Specification. The value of 63 for **vps_max_layers_minus1** is reserved for future use by ITU-T | ISO/IEC. Although the value of **vps_max_layers_minus1** is required to be less than 63 in this version of this Specification, decoders shall allow the value of **vps_max_layers_minus1** equal to 63 to appear in the syntax.

NOTE 2 – The value of 63 for **vps_max_layers_minus1** may be used to indicate an extended number of layers in a future extension where more than 63 layers in a bitstream need to be supported.

The variable **MaxLayersMinus1** is set equal to $\text{Min}(62, \text{vps_max_layers_minus1})$.

vps_max_sub_layers_minus1 plus 1 specifies the maximum number of temporal sub-layers that may be present in each CVS referring to the VPS. The value of **vps_max_sub_layers_minus1** shall be in the range of 0 to 6, inclusive.

vps_temporal_id_nesting_flag, when **vps_max_sub_layers_minus1** is greater than 0, specifies whether inter prediction is additionally restricted for CVSs referring to the VPS. When **vps_max_sub_layers_minus1** is equal to 0, **vps_temporal_id_nesting_flag** shall be equal to 1.

NOTE 3 – The syntax element **vps_temporal_id_nesting_flag** is used to indicate that temporal sub-layer up-switching, i.e., switching from decoding of up to any **TemporalId** **tIdN** to decoding up to any **TemporalId** **tIdM** that is greater than **tIdN**, is always possible.

vps_reserved_0xffff_16bits shall be equal to 0xFFFF in bitstreams conforming to this version of this Specification. Other values for **vps_reserved_0xffff_16bits** are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of **vps_reserved_0xffff_16bits**.

vps_sub_layer_ordering_info_present_flag equal to 1 specifies that **vps_max_dec_pic_buffering_minus1[i]**, **vps_max_num_reorder_pics[i]** and **vps_max_latency_increase_plus1[i]** are present for **vps_max_sub_layers_minus1 + 1** sub-layers. **vps_sub_layer_ordering_info_present_flag** equal to 0 specifies that the values of **vps_max_dec_pic_buffering_minus1[vps_max_sub_layers_minus1]**, **vps_max_num_reorder_pics[vps_max_sub_layers_minus1]** and **vps_max_latency_increase_plus1[vps_max_sub_layers_minus1]** apply to all sub-layers. When **vps_base_layer_internal_flag** is equal to 0, **vps_sub_layer_ordering_info_present_flag** shall be equal to 0 and decoders shall ignore the value of **vps_sub_layer_ordering_info_present_flag**.

vps_max_dec_pic_buffering_minus1[i] plus 1 specifies the maximum required size of the decoded picture buffer for the CVS in units of picture storage buffers when **HighestTid** is equal to **i**. The value of **vps_max_dec_pic_buffering_minus1[i]** shall be in the range of 0 to **MaxDpbSize - 1** (as specified in clause A.4), inclusive. When **i** is greater than 0, **vps_max_dec_pic_buffering_minus1[i]** shall be greater than or equal to **vps_max_dec_pic_buffering_minus1[i - 1]**. When **vps_max_dec_pic_buffering_minus1[i]** is not present for **i** in the range of 0 to **vps_max_sub_layers_minus1 - 1**, inclusive, due to **vps_sub_layer_ordering_info_present_flag** being equal to 0, it is inferred to be equal to **vps_max_dec_pic_buffering_minus1[vps_max_sub_layers_minus1]**. When **vps_base_layer_internal_flag** is equal to 0, **vps_max_dec_pic_buffering_minus1[i]** shall be equal to 0 and decoders shall ignore the value of **vps_max_dec_pic_buffering_minus1[i]**.

vps_max_num_reorder_pics[i] indicates the maximum allowed number of pictures with **PicOutputFlag** equal to 1 that can precede any picture with **PicOutputFlag** equal to 1 in the CVS in decoding order and follow that picture with **PicOutputFlag** equal to 1 in output order when **HighestTid** is equal to **i**. The value of **vps_max_num_reorder_pics[i]** shall be in the range of 0 to **vps_max_dec_pic_buffering_minus1[i]**, inclusive. When **i** is greater than 0, **vps_max_num_reorder_pics[i]** shall be greater than or equal to **vps_max_num_reorder_pics[i - 1]**. When **vps_max_num_reorder_pics[i]** is not present for **i** in the range of 0 to **vps_max_sub_layers_minus1 - 1**, inclusive, due to **vps_sub_layer_ordering_info_present_flag** being equal to 0, it is inferred to be equal to **vps_max_num_reorder_pics[vps_max_sub_layers_minus1]**. When **vps_base_layer_internal_flag** is equal to 0, **vps_max_num_reorder_pics[i]** shall be equal to 0 and decoders shall ignore the value of **vps_max_num_reorder_pics[i]**.

vps_max_latency_increase_plus1[i] not equal to 0 is used to compute the value of **VpsMaxLatencyPictures**[i], which specifies the maximum number of pictures with **PicOutputFlag** equal to 1 that can precede any picture with **PicOutputFlag** equal to 1 in the CVS in output order and follow that picture with **PicOutputFlag** equal to 1 in decoding order when **HighestTid** is equal to i.

When **vps_max_latency_increase_plus1**[i] is not equal to 0, the value of **VpsMaxLatencyPictures**[i] is specified as follows:

$$\text{VpsMaxLatencyPictures}[i] = \text{vps_max_num_reorder_pics}[i] + \text{vps_max_latency_increase_plus1}[i] - 1 \quad (7-2)$$

When **vps_max_latency_increase_plus1**[i] is equal to 0, no corresponding limit is expressed.

The value of **vps_max_latency_increase_plus1**[i] shall be in the range of 0 to $2^{32} - 2$, inclusive. When **vps_max_latency_increase_plus1**[i] is not present for i in the range of 0 to **vps_max_sub_layers_minus1** - 1, inclusive, due to **vps_sub_layer_ordering_info_present_flag** being equal to 0, it is inferred to be equal to **vps_max_latency_increase_plus1**[**vps_max_sub_layers_minus1**].

When **vps_base_layer_internal_flag** is equal to 0, **vps_max_latency_increase_plus1**[i] shall be equal to 0 and decoders shall ignore the value of **vps_max_latency_increase_plus1**[i].

vps_max_layer_id specifies the maximum allowed value of **nuh_layer_id** of all NAL units in each CVS referring to the VPS. **vps_max_layer_id** shall be less than 63 in bitstreams conforming to this version of this Specification. The value of 63 for **vps_max_layer_id** is reserved for future use by ITU-T | ISO/IEC. Although the value of **vps_max_layer_id** is required to be less than 63 in this version of this Specification, decoders shall allow a value of **vps_max_layer_id** equal to 63 to appear in the syntax.

vps_num_layer_sets_minus1 plus 1 specifies the number of layer sets that are specified by the VPS. The value of **vps_num_layer_sets_minus1** shall be in the range of 0 to 1 023, inclusive.

layer_id_included_flag[i][j] equal to 1 specifies that the value of **nuh_layer_id** equal to j is included in the layer identifier list **LayerSetLayerIdList**[i]. **layer_id_included_flag**[i][j] equal to 0 specifies that the value of **nuh_layer_id** equal to j is not included in the layer identifier list **LayerSetLayerIdList**[i].

The value of **NumLayersInIdList**[0] is set equal to 1 and the value of **LayerSetLayerIdList**[0][0] is set equal to 0.

For each value of i in the range of 1 to **vps_num_layer_sets_minus1**, inclusive, the variable **NumLayersInIdList**[i] and the layer identifier list **LayerSetLayerIdList**[i] are derived as follows:

$$\begin{aligned} n &= 0 \\ \text{for}(m = 0; m \leq \text{vps_max_layer_id}; m++) \\ &\quad \text{if}(\text{layer_id_included_flag}[i][m]) \\ &\quad \quad \text{LayerSetLayerIdList}[i][n++] = m \\ \text{NumLayersInIdList}[i] &= n \end{aligned} \quad (7-3)$$

For each value of i in the range of 1 to **vps_num_layer_sets_minus1**, inclusive, **NumLayersInIdList**[i] shall be in the range of 1 to **vps_max_sub_layers_minus1** + 1, inclusive.

When **NumLayersInIdList**[iA] is equal to **NumLayersInIdList**[iB] for any iA and iB in the range of 0 to **vps_num_layer_sets_minus1**, inclusive, with iA not equal to iB, the value of **LayerSetLayerIdList**[iA][n] shall not be equal to **LayerSetLayerIdList**[iB][n] for at least one value of n in the range of 0 to **NumLayersInIdList**[iA], inclusive.

A layer set is identified by the associated layer identifier list. The i-th layer set specified by the VPS is associated with the layer identifier list **LayerSetLayerIdList**[i], for i in the range of 0 to **vps_num_layer_sets_minus1**, inclusive.

A layer set consists of all operation points that are associated with the same layer identifier list.

Each operation point is identified by the associated layer identifier list, denoted as **OpLayerIdList**, which consists of the list of **nuh_layer_id** values of all NAL units included in the operation point, in increasing order of **nuh_layer_id** values, and a variable **OpTid**, which is equal to the highest **TemporalId** of all NAL units included in the operation point. The bitstream subset associated with the operation point identified by **OpLayerIdList** and **OpTid** is the output of the sub-bitstream extraction process as specified in clause 10 with the bitstream, the target highest **TemporalId** equal to **OpTid**, and the target layer identifier list equal to **OpLayerIdList** as inputs. The **OpLayerIdList** and **OpTid** that identify an operation point are also referred to as the **OpLayerIdList** and **OpTid** associated with the operation point, respectively.

vps_timing_info_present_flag equal to 1 specifies that **vps_num_units_in_tick**, **vps_time_scale**, **vps_poc_proportional_to_timing_flag** and **vps_num_hrd_parameters** are present in the VPS.

vps_timing_info_present_flag equal to 0 specifies that **vps_num_units_in_tick**, **vps_time_scale**, **vps_poc_proportional_to_timing_flag** and **vps_num_hrd_parameters** are not present in the VPS.

vps_num_units_in_tick is the number of time units of a clock operating at the frequency **vps_time_scale** Hz that corresponds to one increment (called a clock tick) of a clock tick counter. The value of **vps_num_units_in_tick** shall be greater than 0. A clock tick, in units of seconds, is equal to the quotient of **vps_num_units_in_tick** divided by **vps_time_scale**. For example, when the picture rate of a video signal is 25 Hz, **vps_time_scale** may be equal to 27 000 000 and **vps_num_units_in_tick** may be equal to 1 080 000, and consequently a clock tick may be 0.04 seconds.

vps_time_scale is the number of time units that pass in one second. For example, a time coordinate system that measures time using a 27 MHz clock has a **vps_time_scale** of 27 000 000. The value of **vps_time_scale** shall be greater than 0.

vps_poc_proportional_to_timing_flag equal to 1 indicates that the picture order count value for each picture in the CVS that is not the first picture in the CVS, in decoding order, is proportional to the output time of the picture relative to the output time of the first picture in the CVS. **vps_poc_proportional_to_timing_flag** equal to 0 indicates that the picture order count value for each picture in the CVS that is not the first picture in the CVS, in decoding order, may or may not be proportional to the output time of the picture relative to the output time of the first picture in the CVS.

vps_num_ticks_poc_diff_one_minus1 plus 1 specifies the number of clock ticks corresponding to a difference of picture order count values equal to 1. The value of **vps_num_ticks_poc_diff_one_minus1** shall be in the range of 0 to $2^{32} - 2$, inclusive.

vps_num_hrd_parameters specifies the number of **hrd_parameters()** syntax structures present in the VPS RBSP before the **vps_extension_flag** syntax element. The value of **vps_num_hrd_parameters** shall be in the range of 0 to **vps_num_layer_sets_minus1** + 1, inclusive.

hrd_layer_set_idx[i] specifies the index, into the list of layer sets specified by the VPS, of the layer set to which the *i*-th **hrd_parameters()** syntax structure in the VPS applies. The value of **hrd_layer_set_idx[i]** shall be in the range of (**vps_base_layer_internal_flag** ? 0 : 1) to **vps_num_layer_sets_minus1**, inclusive.

It is a requirement of bitstream conformance that the value of **hrd_layer_set_idx[i]** shall not be equal to the value of **hrd_layer_set_idx[j]** for any value of *j* not equal to *i*.

cprms_present_flag[i] equal to 1 specifies that the HRD parameters that are common for all sub-layers are present in the *i*-th **hrd_parameters()** syntax structure in the VPS. **cprms_present_flag[i]** equal to 0 specifies that the HRD parameters that are common for all sub-layers are not present in the *i*-th **hrd_parameters()** syntax structure in the VPS and are derived to be the same as the (*i* - 1)-th **hrd_parameters()** syntax structure in the VPS. **cprms_present_flag[0]** is inferred to be equal to 1.

vps_extension_flag equal to 0 specifies that no **vps_extension_data_flag** syntax elements are present in the VPS RBSP syntax structure. **vps_extension_flag** equal to 1 specifies that there are **vps_extension_data_flag** syntax elements present in the VPS RBSP syntax structure. Decoders conforming to a profile specified in Annex A but not supporting the INBLD capability specified in Annex F shall ignore all data that follow the value 1 for **vps_extension_flag** in a VPS NAL unit.

vps_extension_data_flag may have any value. Its presence and value do not affect the decoding process of the profiles specified in Annex A. Decoders conforming to a profile specified in Annex A but not supporting the INBLD capability specified in Annex F shall ignore all **vps_extension_data_flag** syntax elements.

7.4.3.2 Sequence parameter set RBSP semantics

7.4.3.2.1 General sequence parameter set RBSP semantics

sps_video_parameter_set_id specifies the value of the **vps_video_parameter_set_id** of the active VPS.

sps_max_sub_layers_minus1 plus 1 specifies the maximum number of temporal sub-layers that may be present in each CVS referring to the SPS. The value of **sps_max_sub_layers_minus1** shall be in the range of 0 to 6, inclusive. The value of **sps_max_sub_layers_minus1** shall be less than or equal to **vps_max_sub_layers_minus1**.

sps_temporal_id_nesting_flag, when **sps_max_sub_layers_minus1** is greater than 0, specifies whether inter prediction is additionally restricted for CVSs referring to the SPS. When **vps_temporal_id_nesting_flag** is equal to 1, **sps_temporal_id_nesting_flag** shall be equal to 1. When **sps_max_sub_layers_minus1** is equal to 0, **sps_temporal_id_nesting_flag** shall be equal to 1.

NOTE 1 – The syntax element **sps_temporal_id_nesting_flag** is used to indicate that temporal up-switching, i.e., switching from decoding up to any TemporalId tldN to decoding up to any TemporalId tldM that is greater than tldN, is always possible in the CVS.

sps_seq_parameter_set_id provides an identifier for the SPS for reference by other syntax elements. The value of **sps_seq_parameter_set_id** shall be in the range of 0 to 15, inclusive.

chroma_format_idc specifies the chroma sampling relative to the luma sampling as specified in clause 6.2. The value of **chroma_format_idc** shall be in the range of 0 to 3, inclusive.

separate_colour_plane_flag equal to 1 specifies that the three colour components of the 4:4:4 chroma format are coded separately. **separate_colour_plane_flag** equal to 0 specifies that the colour components are not coded separately. When **separate_colour_plane_flag** is not present, it is inferred to be equal to 0. When **separate_colour_plane_flag** is equal to 1, the coded picture consists of three separate components, each of which consists of coded samples of one colour plane (Y, Cb, or Cr) and uses the monochrome coding syntax. In this case, each colour plane is associated with a specific **colour_plane_id** value.

NOTE 2 – There is no dependency in decoding processes between the colour planes having different **colour_plane_id** values. For example, the decoding process of a monochrome picture with one value of **colour_plane_id** does not use any data from monochrome pictures having different values of **colour_plane_id** for inter prediction.

Depending on the value of **separate_colour_plane_flag**, the value of the variable **ChromaArrayType** is assigned as follows:

- If **separate_colour_plane_flag** is equal to 0, **ChromaArrayType** is set equal to **chroma_format_idc**.
- Otherwise (**separate_colour_plane_flag** is equal to 1), **ChromaArrayType** is set equal to 0.

pic_width_in_luma_samples specifies the width of each decoded picture in units of luma samples. **pic_width_in_luma_samples** shall not be equal to 0 and shall be an integer multiple of **MinCbSizeY**.

pic_height_in_luma_samples specifies the height of each decoded picture in units of luma samples. **pic_height_in_luma_samples** shall not be equal to 0 and shall be an integer multiple of **MinCbSizeY**.

conformance_window_flag equal to 1 indicates that the conformance cropping window offset parameters follow next in the SPS. **conformance_window_flag** equal to 0 indicates that the conformance cropping window offset parameters are not present.

conf_win_left_offset, **conf_win_right_offset**, **conf_win_top_offset** and **conf_win_bottom_offset** specify the samples of the pictures in the CVS that are output from the decoding process, in terms of a rectangular region specified in picture coordinates for output. When **conformance_window_flag** is equal to 0, the values of **conf_win_left_offset**, **conf_win_right_offset**, **conf_win_top_offset** and **conf_win_bottom_offset** are inferred to be equal to 0.

The conformance cropping window contains the luma samples with horizontal picture coordinates from $\text{SubWidthC} * \text{conf_win_left_offset}$ to $\text{pic_width_in_luma_samples} - (\text{SubWidthC} * \text{conf_win_right_offset} + 1)$ and vertical picture coordinates from $\text{SubHeightC} * \text{conf_win_top_offset}$ to $\text{pic_height_in_luma_samples} - (\text{SubHeightC} * \text{conf_win_bottom_offset} + 1)$, inclusive.

The value of $\text{SubWidthC} * (\text{conf_win_left_offset} + \text{conf_win_right_offset})$ shall be less than **pic_width_in_luma_samples**, and the value of $\text{SubHeightC} * (\text{conf_win_top_offset} + \text{conf_win_bottom_offset})$ shall be less than **pic_height_in_luma_samples**.

When **ChromaArrayType** is not equal to 0, the corresponding specified samples of the two chroma arrays are the samples having picture coordinates ($x / \text{SubWidthC}$, $y / \text{SubHeightC}$), where (x , y) are the picture coordinates of the specified luma samples.

NOTE 3 – The conformance cropping window offset parameters are only applied at the output. All internal decoding processes are applied to the uncropped picture size.

bit_depth_luma_minus8 specifies the bit depth of the samples of the luma array **BitDepth_Y** and the value of the luma quantization parameter range offset **QpBdOffset_Y** as follows:

$$\text{BitDepth}_Y = 8 + \text{bit_depth_luma_minus8} \quad (7-4)$$

$$\text{QpBdOffset}_Y = 6 * \text{bit_depth_luma_minus8} \quad (7-5)$$

bit_depth_luma_minus8 shall be in the range of 0 to 8, inclusive.

bit_depth_chroma_minus8 specifies the bit depth of the samples of the chroma arrays **BitDepth_C** and the value of the chroma quantization parameter range offset **QpBdOffset_C** as follows:

$$\text{BitDepth}_C = 8 + \text{bit_depth_chroma_minus8} \quad (7-6)$$

$$\text{QpBdOffset}_C = 6 * \text{bit_depth_chroma_minus8} \quad (7-7)$$

bit_depth_chroma_minus8 shall be in the range of 0 to 8, inclusive.

log2_max_pic_order_cnt_lsb_minus4 specifies the value of the variable **MaxPicOrderCntLsb** that is used in the decoding process for picture order count as follows:

$$\text{MaxPicOrderCntLsb} = 2^{(\log_2 \text{max_pic_order_cnt_lsb_minus4} + 4)} \quad (7-8)$$

The value of `log2_max_pic_order_cnt_lsb_minus4` shall be in the range of 0 to 12, inclusive.

`sps_sub_layer_ordering_info_present_flag` equal to 1 specifies that `sps_max_dec_pic_buffering_minus1[i]`, `sps_max_num_reorder_pics[i]` and `sps_max_latency_increase_plus1[i]` are present for `sps_max_sub_layers_minus1 + 1` sub-layers. `sps_sub_layer_ordering_info_present_flag` equal to 0 specifies that the values of `sps_max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1]`, `sps_max_num_reorder_pics[sps_max_sub_layers_minus1]` and `sps_max_latency_increase_plus1[sps_max_sub_layers_minus1]` apply to all sub-layers.

`sps_max_dec_pic_buffering_minus1[i]` plus 1 specifies the maximum required size of the decoded picture buffer for the CVS in units of picture storage buffers when `HighestTid` is equal to `i`. The value of `sps_max_dec_pic_buffering_minus1[i]` shall be in the range of 0 to `MaxDpbSize - 1`, inclusive, where `MaxDpbSize` is as specified in clause A.4. When `i` is greater than 0, `sps_max_dec_pic_buffering_minus1[i]` shall be greater than or equal to `sps_max_dec_pic_buffering_minus1[i - 1]`. The value of `sps_max_dec_pic_buffering_minus1[i]` shall be less than or equal to `vps_max_dec_pic_buffering_minus1[i]` for each value of `i`. When `sps_max_dec_pic_buffering_minus1[i]` is not present for `i` in the range of 0 to `sps_max_sub_layers_minus1 - 1`, inclusive, due to `sps_sub_layer_ordering_info_present_flag` being equal to 0, it is inferred to be equal to `sps_max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1]`.

`sps_max_num_reorder_pics[i]` indicates the maximum allowed number of pictures with `PicOutputFlag` equal to 1 that can precede any picture with `PicOutputFlag` equal to 1 in the CVS in decoding order and follow that picture with `PicOutputFlag` equal to 1 in output order when `HighestTid` is equal to `i`. The value of `sps_max_num_reorder_pics[i]` shall be in the range of 0 to `sps_max_dec_pic_buffering_minus1[i]`, inclusive. When `i` is greater than 0, `sps_max_num_reorder_pics[i]` shall be greater than or equal to `sps_max_num_reorder_pics[i - 1]`. The value of `sps_max_num_reorder_pics[i]` shall be less than or equal to `vps_max_num_reorder_pics[i]` for each value of `i`. When `sps_max_num_reorder_pics[i]` is not present for `i` in the range of 0 to `sps_max_sub_layers_minus1 - 1`, inclusive, due to `sps_sub_layer_ordering_info_present_flag` being equal to 0, it is inferred to be equal to `sps_max_num_reorder_pics[sps_max_sub_layers_minus1]`.

`sps_max_latency_increase_plus1[i]` not equal to 0 is used to compute the value of `SpsMaxLatencyPictures[i]`, which specifies the maximum number of pictures with `PicOutputFlag` equal to 1 that can precede any picture with `PicOutputFlag` equal to 1 in the CVS in output order and follow that picture with `PicOutputFlag` equal to 1 in decoding order when `HighestTid` is equal to `i`.

When `sps_max_latency_increase_plus1[i]` is not equal to 0, the value of `SpsMaxLatencyPictures[i]` is specified as follows:

$$\text{SpsMaxLatencyPictures}[i] = \text{sps_max_num_reorder_pics}[i] + \text{sps_max_latency_increase_plus1}[i] - 1 \quad (7-9)$$

When `sps_max_latency_increase_plus1[i]` is equal to 0, no corresponding limit is expressed.

The value of `sps_max_latency_increase_plus1[i]` shall be in the range of 0 to $2^{32} - 2$, inclusive. When `vps_max_latency_increase_plus1[i]` is not equal to 0, the value of `sps_max_latency_increase_plus1[i]` shall not be equal to 0 and shall be less than or equal to `vps_max_latency_increase_plus1[i]` for each value of `i`. When `sps_max_latency_increase_plus1[i]` is not present for `i` in the range of 0 to `sps_max_sub_layers_minus1 - 1`, inclusive, due to `sps_sub_layer_ordering_info_present_flag` being equal to 0, it is inferred to be equal to `sps_max_latency_increase_plus1[sps_max_sub_layers_minus1]`.

`log2_min_luma_coding_block_size_minus3` plus 3 specifies the minimum luma coding block size.

`log2_diff_max_min_luma_coding_block_size` specifies the difference between the maximum and minimum luma coding block size.

The variables `MinCbLog2SizeY`, `CtbLog2SizeY`, `MinCbSizeY`, `CtbSizeY`, `PicWidthInMinCbsY`, `PicWidthInCtbsY`, `PicHeightInMinCbsY`, `PicHeightInCtbsY`, `PicSizeInMinCbsY`, `PicSizeInCtbsY`, `PicSizeInSamplesY`, `PicWidthInSamplesC` and `PicHeightInSamplesC` are derived as follows:

$$\text{MinCbLog2SizeY} = \text{log2_min_luma_coding_block_size_minus3} + 3 \quad (7-10)$$

$$\text{CtbLog2SizeY} = \text{MinCbLog2SizeY} + \text{log2_diff_max_min_luma_coding_block_size} \quad (7-11)$$

$$\text{MinCbSizeY} = 1 \ll \text{MinCbLog2SizeY} \quad (7-12)$$

$$\text{CtbSizeY} = 1 \ll \text{CtbLog2SizeY} \quad (7-13)$$

$$\text{PicWidthInMinCbsY} = \text{pic_width_in_luma_samples} / \text{MinCbSizeY} \quad (7-14)$$

$$\text{PicWidthInCtbsY} = \text{Ceil}(\text{pic_width_in_luma_samples} \div \text{CtbSizeY}) \quad (7-15)$$

$$\text{PicHeightInMinCbsY} = \text{pic_height_in_luma_samples} / \text{MinCbSizeY} \quad (7-16)$$

$$\text{PicHeightInCtbsY} = \text{Ceil}(\text{pic_height_in_luma_samples} \div \text{CtbSizeY}) \quad (7-17)$$

$$\text{PicSizeInMinCbsY} = \text{PicWidthInMinCbsY} * \text{PicHeightInMinCbsY} \quad (7-18)$$

$$\text{PicSizeInCtbsY} = \text{PicWidthInCtbsY} * \text{PicHeightInCtbsY} \quad (7-19)$$

$$\text{PicSizeInSamplesY} = \text{pic_width_in_luma_samples} * \text{pic_height_in_luma_samples} \quad (7-20)$$

$$\text{PicWidthInSamplesC} = \text{pic_width_in_luma_samples} / \text{SubWidthC} \quad (7-21)$$

$$\text{PicHeightInSamplesC} = \text{pic_height_in_luma_samples} / \text{SubHeightC} \quad (7-22)$$

The variables CtbWidthC and CtbHeightC, which specify the width and height, respectively, of the array for each chroma CTB, are derived as follows:

- If chroma_format_idc is equal to 0 (monochrome) or separate_colour_plane_flag is equal to 1, CtbWidthC and CtbHeightC are both equal to 0.
- Otherwise, CtbWidthC and CtbHeightC are derived as follows:

$$\text{CtbWidthC} = \text{CtbSizeY} / \text{SubWidthC} \quad (7-23)$$

$$\text{CtbHeightC} = \text{CtbSizeY} / \text{SubHeightC} \quad (7-24)$$

log2_min_luma_transform_block_size_minus2 plus 2 specifies the minimum luma transform block size.

The variable MinTbLog2SizeY is set equal to log2_min_luma_transform_block_size_minus2 + 2. The CVS shall not contain data that result in MinTbLog2SizeY greater than or equal to MinCbLog2SizeY.

log2_diff_max_min_luma_transform_block_size specifies the difference between the maximum and minimum luma transform block size.

The variable MaxTbLog2SizeY is set equal to log2_min_luma_transform_block_size_minus2 + 2 + log2_diff_max_min_luma_transform_block_size.

The CVS shall not contain data that result in MaxTbLog2SizeY greater than Min(CtbLog2SizeY, 5).

The array ScanOrder[log2BlockSize][scanIdx][sPos][sComp] specifies the mapping of the scan position sPos, ranging from 0 to (1 << log2BlockSize) * (1 << log2BlockSize) – 1, inclusive, to horizontal and vertical components of the scan-order matrix. The array index scanIdx equal to 0 specifies an up-right diagonal scan order, scanIdx equal to 1 specifies a horizontal scan order, scanIdx equal to 2 specifies a vertical scan order, and scanIdx equal to 3 specifies a traverse scan order. The array index sComp equal to 0 specifies the horizontal component and the array index sComp equal to 1 specifies the vertical component. The array ScanOrder is derived as follows:

For the variable log2BlockSize ranging from 0 to 5, inclusive, the scanning order array ScanOrder is derived as follows:

- For log2BlockSize ranging from 0 to 3, inclusive, the up-right diagonal scan order array initialization process as specified in clause 6.5.3 is invoked with 1 << log2BlockSize as input, and the output is assigned to ScanOrder[log2BlockSize][0].
- For log2BlockSize ranging from 0 to 3, inclusive, the horizontal scan order array initialization process as specified in clause 6.5.4 is invoked with 1 << log2BlockSize as input, and the output is assigned to ScanOrder[log2BlockSize][1].
- For log2BlockSize ranging from 0 to 3, inclusive, the vertical scan order array initialization process as specified in clause 6.5.5 is invoked with 1 << log2BlockSize as input, and the output is assigned to ScanOrder[log2BlockSize][2].
- For log2BlockSize ranging from 2 to 5, inclusive, the traverse scan order array initialization process as specified in clause 6.5.6 is invoked with 1 << log2BlockSize as input, and the output is assigned to ScanOrder[log2BlockSize][3].

max_transform_hierarchy_depth_inter specifies the maximum hierarchy depth for transform units of coding units coded in inter prediction mode. The value of **max_transform_hierarchy_depth_inter** shall be in the range of 0 to $\text{CtbLog2SizeY} - \text{MinTbLog2SizeY}$, inclusive.

max_transform_hierarchy_depth_intra specifies the maximum hierarchy depth for transform units of coding units coded in intra prediction mode. The value of **max_transform_hierarchy_depth_intra** shall be in the range of 0 to $\text{CtbLog2SizeY} - \text{MinTbLog2SizeY}$, inclusive.

scaling_list_enabled_flag equal to 1 specifies that a scaling list is used for the scaling process for transform coefficients. **scaling_list_enabled_flag** equal to 0 specifies that scaling list is not used for the scaling process for transform coefficients.

sps_scaling_list_data_present_flag equal to 1 specifies that the **scaling_list_data()** syntax structure is present in the SPS. **sps_scaling_list_data_present_flag** equal to 0 specifies that the **scaling_list_data()** syntax structure is not present in the SPS. When not present, the value of **sps_scaling_list_data_present_flag** is inferred to be equal to 0.

amp_enabled_flag equal to 1 specifies that asymmetric motion partitions, i.e., **PartMode** equal to **PART_2NxN**, **PART_2NxN_D**, **PART_nLx2N** or **PART_nRx2N**, may be used in CTBs. **amp_enabled_flag** equal to 0 specifies that asymmetric motion partitions cannot be used in CTBs.

sample_adaptive_offset_enabled_flag equal to 1 specifies that the sample adaptive offset process is applied to the reconstructed picture after the deblocking filter process. **sample_adaptive_offset_enabled_flag** equal to 0 specifies that the sample adaptive offset process is not applied to the reconstructed picture after the deblocking filter process.

pcm_enabled_flag equal to 0 specifies that PCM-related syntax (**pcm_sample_bit_depth_luma_minus1**, **pcm_sample_bit_depth_chroma_minus1**, **log2_min_pcm_luma_coding_block_size_minus3**, **log2_diff_max_min_pcm_luma_coding_block_size**, **pcm_loop_filter_disabled_flag**, **pcm_flag**, **pcm_alignment_zero_bit** syntax elements and **pcm_sample()** syntax structure) is not present in the CVS.

NOTE 4 – When **MinCbLog2SizeY** is equal to 6 and **pcm_enabled_flag** is equal to 1, PCM sample data-related syntax (**pcm_flag**, **pcm_alignment_zero_bit** syntax elements and **pcm_sample()** syntax structure) is not present in the CVS, because the maximum size of coding blocks that can convey PCM sample data-related syntax is restricted to be less than or equal to $\text{Min}(\text{CtbLog2SizeY}, 5)$. Hence, **MinCbLog2SizeY** equal to 6 with **pcm_enabled_flag** equal to 1 is not an appropriate setting to convey PCM sample data in the CVS.

pcm_sample_bit_depth_luma_minus1 specifies the number of bits used to represent each of PCM sample values of the luma component as follows:

$$\text{PcmBitDepth}_Y = \text{pcm_sample_bit_depth_luma_minus1} + 1 \quad (7-25)$$

The value of PcmBitDepth_Y shall be less than or equal to the value of BitDepth_Y .

pcm_sample_bit_depth_chroma_minus1 specifies the number of bits used to represent each of PCM sample values of the chroma components as follows:

$$\text{PcmBitDepth}_C = \text{pcm_sample_bit_depth_chroma_minus1} + 1 \quad (7-26)$$

The value of PcmBitDepth_C shall be less than or equal to the value of BitDepth_C . When **ChromaArrayType** is equal to 0, **pcm_sample_bit_depth_chroma_minus1** is not used in the decoding process and decoders shall ignore its value.

log2_min_pcm_luma_coding_block_size_minus3 plus 3 specifies the minimum size of coding blocks with **pcm_flag** equal to 1.

The variable $\text{Log2MinIpcmCbSizeY}$ is set equal to **log2_min_pcm_luma_coding_block_size_minus3** + 3. The value of $\text{Log2MinIpcmCbSizeY}$ shall be in the range of $\text{Min}(\text{MinCbLog2SizeY}, 5)$ to $\text{Min}(\text{CtbLog2SizeY}, 5)$, inclusive.

log2_diff_max_min_pcm_luma_coding_block_size specifies the difference between the maximum and minimum size of coding blocks with **pcm_flag** equal to 1.

The variable $\text{Log2MaxIpcmCbSizeY}$ is set equal to **log2_diff_max_min_pcm_luma_coding_block_size** + $\text{Log2MinIpcmCbSizeY}$. The value of $\text{Log2MaxIpcmCbSizeY}$ shall be less than or equal to $\text{Min}(\text{CtbLog2SizeY}, 5)$.

pcm_loop_filter_disabled_flag specifies whether the loop filter process is disabled on reconstructed samples in a coding unit with **pcm_flag** equal to 1 as follows:

- If **pcm_loop_filter_disabled_flag** is equal to 1, the deblocking filter and sample adaptive offset filter processes on the reconstructed samples in a coding unit with **pcm_flag** equal to 1 are disabled.
- Otherwise (**pcm_loop_filter_disabled_flag** value is equal to 0), the deblocking filter and sample adaptive offset filter processes on the reconstructed samples in a coding unit with **pcm_flag** equal to 1 are not disabled.

When **pcm_loop_filter_disabled_flag** is not present, it is inferred to be equal to 0.

num_short_term_ref_pic_sets specifies the number of `st_ref_pic_set()` syntax structures included in the SPS. The value of `num_short_term_ref_pic_sets` shall be in the range of 0 to 64, inclusive.

NOTE 5 – A decoder should allocate memory for a total number of `num_short_term_ref_pic_sets + 1` `st_ref_pic_set()` syntax structures since there may be a `st_ref_pic_set()` syntax structure directly signalled in the slice headers of a current picture. A `st_ref_pic_set()` syntax structure directly signalled in the slice headers of a current picture has an index equal to `num_short_term_ref_pic_sets`.

long_term_ref_pics_present_flag equal to 0 specifies that no long-term reference picture is used for inter prediction of any coded picture in the CVS. **long_term_ref_pics_present_flag** equal to 1 specifies that long-term reference pictures may be used for inter prediction of one or more coded pictures in the CVS.

num_long_term_ref_pics_sps specifies the number of candidate long-term reference pictures that are specified in the SPS. The value of `num_long_term_ref_pics_sps` shall be in the range of 0 to 32, inclusive.

lt_ref_pic_poc_lsb_sps[i] specifies the picture order count modulo `MaxPicOrderCntLsb` of the *i*-th candidate long-term reference picture specified in the SPS. The number of bits used to represent `lt_ref_pic_poc_lsb_sps[i]` is equal to `log2_max_pic_order_cnt_lsb_minus4 + 4`.

used_by_curr_pic_lt_sps_flag[i] equal to 0 specifies that the *i*-th candidate long-term reference picture specified in the SPS is not used for reference by a picture that includes in its long-term reference picture set (RPS) the *i*-th candidate long-term reference picture specified in the SPS.

sps_temporal_mvp_enabled_flag equal to 1 specifies that `slice_temporal_mvp_enabled_flag` is present in the slice headers of non-IDR pictures in the CVS. **sps_temporal_mvp_enabled_flag** equal to 0 specifies that `slice_temporal_mvp_enabled_flag` is not present in slice headers and that temporal motion vector predictors are not used in the CVS.

strong_intra_smoothing_enabled_flag equal to 1 specifies that bi-linear interpolation is conditionally used in the intra prediction filtering process in the CVS as specified in clause 8.4.4.2.3. **strong_intra_smoothing_enabled_flag** equal to 0 specifies that the bi-linear interpolation is not used in the CVS.

vui_parameters_present_flag equal to 1 specifies that the `vui_parameters()` syntax structure as specified in Annex E is present. **vui_parameters_present_flag** equal to 0 specifies that the `vui_parameters()` syntax structure as specified in Annex E is not present.

sps_extension_present_flag equal to 1 specifies that the syntax elements `sps_range_extension_flag`, `sps_multilayer_extension_flag`, `sps_3d_extension_flag`, `sps_scc_extension_flag`, and `sps_extension_4bits` are present in the SPS RBSP syntax structure. **sps_extension_present_flag** equal to 0 specifies that these syntax elements are not present.

sps_range_extension_flag equal to 1 specifies that the `sps_range_extension()` syntax structure is present in the SPS RBSP syntax structure. **sps_range_extension_flag** equal to 0 specifies that this syntax structure is not present. When not present, the value of `sps_range_extension_flag` is inferred to be equal to 0.

sps_multilayer_extension_flag equal to 1 specifies that the `sps_multilayer_extension()` syntax structure (specified in Annex F) is present in the SPS RBSP syntax structure. **sps_multilayer_extension_flag** equal to 0 specifies that the `sps_multilayer_extension()` syntax structure is not present. When not present, the value of `sps_multilayer_extension_flag` is inferred to be equal to 0.

sps_3d_extension_flag equal to 1 specifies that the `sps_3d_extension()` syntax structure (specified in Annex I) is present in the SPS RBSP syntax structure. **sps_3d_extension_flag** equal to 0 specifies that the `sps_3d_extension()` syntax structure is not present. When not present, the value of `sps_3d_extension_flag` is inferred to be equal to 0.

sps_scc_extension_flag equal to 1 specifies that the `sps_scc_extension()` syntax structure is present in the SPS RBSP syntax structure. **sps_scc_extension_flag** equal to 0 specifies that this syntax structure is not present. When not present, the value of `sps_scc_extension_flag` is inferred to be equal to 0.

sps_extension_4bits equal to 0 specifies that no `sps_extension_data_flag` syntax elements are present in the SPS RBSP syntax structure. When present, `sps_extension_4bits` shall be equal to 0 in bitstreams conforming to this version of this Specification. Values of `sps_extension_4bits` not equal to 0 are reserved for future use by ITU-T | ISO/IEC. Decoders shall allow the value of `sps_extension_4bits` to be not equal to 0 and shall ignore all `sps_extension_data_flag` syntax elements in an SPS NAL unit. When not present, the value of `sps_extension_4bits` is inferred to be equal to 0.

sps_extension_data_flag may have any value. Its presence and value do not affect the decoding process specified in this version of this Specification. Decoders conforming to this version of this Specification shall ignore all `sps_extension_data_flag` syntax elements.

7.4.3.2.2 Sequence parameter set range extension semantics

transform_skip_rotation_enabled_flag equal to 1 specifies that a rotation is applied to the residual data block for intra 4x4 blocks coded using a transform skip operation. **transform_skip_rotation_enabled_flag** equal to 0 specifies that this rotation is not applied. When not present, the value of **transform_skip_rotation_enabled_flag** is inferred to be equal to 0.

transform_skip_context_enabled_flag equal to 1 specifies that a particular context is used for the parsing of the **sig_coeff_flag** for transform blocks with a skipped transform. **transform_skip_context_enabled_flag** equal to 0 specifies that the presence or absence of transform skipping or a transform bypass for transform blocks is not used in the context selection for this flag. When not present, the value of **transform_skip_context_enabled_flag** is inferred to be equal to 0.

implicit_rdp_pcm_enabled_flag equal to 1 specifies that the residual modification process for blocks using a transform bypass may be used for intra blocks in the CVS. **implicit_rdp_pcm_enabled_flag** equal to 0 specifies that the residual modification process is not used for intra blocks in the CVS. When not present, the value of **implicit_rdp_pcm_enabled_flag** is inferred to be equal to 0.

explicit_rdp_pcm_enabled_flag equal to 1 specifies that the residual modification process for blocks using a transform bypass may be used for inter blocks in the CVS. **explicit_rdp_pcm_enabled_flag** equal to 0 specifies that the residual modification process is not used for inter blocks in the CVS. When not present, the value of **explicit_rdp_pcm_enabled_flag** is inferred to be equal to 0.

extended_precision_processing_flag equal to 1 specifies that an extended dynamic range is used for transform coefficients and transform processing. **extended_precision_processing_flag** equal to 0 specifies that the extended dynamic range is not used. When not present, the value of **extended_precision_processing_flag** is inferred to be equal to 0.

The variables CoeffMin_Y , CoeffMin_C , CoeffMax_Y and CoeffMax_C are derived as follows:

$$\text{CoeffMin}_Y = -(1 \ll (\text{extended_precision_processing_flag} ? \text{Max}(15, \text{BitDepth}_Y + 6) : 15)) \quad (7-27)$$

$$\text{CoeffMin}_C = -(1 \ll (\text{extended_precision_processing_flag} ? \text{Max}(15, \text{BitDepth}_C + 6) : 15)) \quad (7-28)$$

$$\text{CoeffMax}_Y = (1 \ll (\text{extended_precision_processing_flag} ? \text{Max}(15, \text{BitDepth}_Y + 6) : 15)) - 1 \quad (7-29)$$

$$\text{CoeffMax}_C = (1 \ll (\text{extended_precision_processing_flag} ? \text{Max}(15, \text{BitDepth}_C + 6) : 15)) - 1 \quad (7-30)$$

intra_smoothing_disabled_flag equal to 1 specifies that the filtering process of neighbouring samples is unconditionally disabled for intra prediction. **intra_smoothing_disabled_flag** equal to 0 specifies that the filtering process of neighbouring samples is not disabled. When not present, the value of **intra_smoothing_disabled_flag** is inferred to be equal to 0.

high_precision_offsets_enabled_flag equal to 1 specifies that weighted prediction offset values are signalled using a bit-depth-dependent precision. **high_precision_offsets_enabled_flag** equal to 0 specifies that weighted prediction offset values are signalled with a precision equivalent to eight bit processing.

The variables WpOffsetBdShift_Y , WpOffsetBdShift_C , $\text{WpOffsetHalfRange}_Y$ and $\text{WpOffsetHalfRange}_C$ are derived as follows:

$$\text{WpOffsetBdShift}_Y = \text{high_precision_offsets_enabled_flag} ? 0 : (\text{BitDepth}_Y - 8) \quad (7-31)$$

$$\text{WpOffsetBdShift}_C = \text{high_precision_offsets_enabled_flag} ? 0 : (\text{BitDepth}_C - 8) \quad (7-32)$$

$$\text{WpOffsetHalfRange}_Y = 1 \ll (\text{high_precision_offsets_enabled_flag} ? (\text{BitDepth}_Y - 1) : 7) \quad (7-33)$$

$$\text{WpOffsetHalfRange}_C = 1 \ll (\text{high_precision_offsets_enabled_flag} ? (\text{BitDepth}_C - 1) : 7) \quad (7-34)$$

persistent_rice_adaptation_enabled_flag equal to 1 specifies that the Rice parameter derivation for the binarization of **coeff_abs_level_remaining[]** is initialized at the start of each sub-block using mode dependent statistics accumulated from previous sub-blocks. **persistent_rice_adaptation_enabled_flag** equal to 0 specifies that no previous sub-block state is used in Rice parameter derivation. When not present, the value of **persistent_rice_adaptation_enabled_flag** is inferred to be equal to 0.

cabac_bypass_alignment_enabled_flag equal to 1 specifies that a context-based adaptive binary arithmetic coding (CABAC) alignment process is used prior to bypass decoding of the syntax elements **coeff_sign_flag[]** and **coeff_abs_level_remaining[]**. **cabac_bypass_alignment_enabled_flag** equal to 0 specifies that no CABAC alignment process is used prior to bypass decoding. When not present, the value of **cabac_bypass_alignment_enabled_flag** is inferred to be equal to 0.

7.4.3.2.3 Sequence parameter set screen content coding extension semantics

sps_curr_pic_ref_enabled_flag equal to 1 specifies that a picture in the CVS may be included in a reference picture list of a slice of the picture itself. **sps_curr_pic_ref_enabled_flag** equal to 0 specifies that a picture in the CVS is never included

in a reference picture list of a slice of the picture itself. When not present, the value of `sps_curr_pic_ref_enabled_flag` is inferred to be equal to 0.

palette_mode_enabled_flag equal to 1 specifies that the decoding process for palette mode may be used for intra blocks. `palette_mode_enabled_flag` equal to 0 specifies that the decoding process for palette mode is not applied. When not present, the value of `palette_mode_enabled_flag` is inferred to be equal to 0.

palette_max_size specifies the maximum allowed palette size. When not present, the value of `palette_max_size` is inferred to be 0.

delta_palette_max_predictor_size specifies the difference between the maximum allowed palette predictor size and the maximum allowed palette size. When not present, the value of `delta_palette_max_predictor_size` is inferred to be 0. The variable `PaletteMaxPredictorSize` is derived as follows:

$$\text{PaletteMaxPredictorSize} = \text{palette_max_size} + \text{delta_palette_max_predictor_size} \quad (7-35)$$

It is a requirement of bitstream conformance that, when `palette_max_size` is equal to 0, the value of `delta_palette_max_predictor_size` shall be equal to 0.

sps_palette_predictor_initializers_present_flag equal to 1 specifies that the sequence palette predictors are initialized using the `sps_palette_predictor_initializers`. `sps_palette_predictor_initializers_present_flag` equal to 0 specifies that the entries in the sequence palette predictor are initialized to 0. When not present, the value of `sps_palette_predictor_initializers_present_flag` is inferred to be equal to 0.

It is a requirement of bitstream conformance that, when `palette_max_size` is equal to 0, the value of `sps_palette_predictor_initializers_present_flag` shall be equal to 0.

sps_num_palette_predictor_initializers_minus1 plus 1 specifies the number of entries in the sequence palette predictor initializer.

It is a requirement of bitstream conformance that the value of `sps_num_palette_predictor_initializers_minus1` plus 1 shall be less than or equal to `PaletteMaxPredictorSize`.

sps_palette_predictor_initializer[comp][i] specifies the value of the `comp`-th component of the `i`-th palette entry in the SPS that is used to initialize the array `PredictorPaletteEntries`. For values of `i` in the range of 0 to `sps_num_palette_predictor_initializers_minus1`, inclusive, the value of the `sps_palette_predictor_initializer[0][i]` shall be in the range of 0 to $(1 \ll \text{BitDepth}_Y) - 1$, inclusive, and the values of `sps_palette_predictor_initializer[1][i]` and `sps_palette_predictor_initializer[2][i]` shall be in the range of 0 to $(1 \ll \text{BitDepth}_C) - 1$, inclusive.

motion_vector_resolution_control_idc controls the presence and inference of the `use_integer_mv_flag` that specifies the resolution of motion vectors for inter prediction. The value of `motion_vector_resolution_control_idc` shall not be equal to 3 in bitstreams conforming to this version of this Specification. The value of 3 for `motion_vector_resolution_control_idc` is reserved for future use by ITU-T | ISO/IEC. When not present, the value of `motion_vector_resolution_control_idc` is inferred to be equal to 0.

intra_boundary_filtering_disabled_flag equal to 1 specifies that the intra boundary filtering process is unconditionally disabled for intra prediction. `intra_boundary_filtering_disabled_flag` equal to 0 specifies that the intra boundary filtering process may be used. When not present, the value of `intra_boundary_filtering_disabled_flag` is inferred to be equal to 0.

7.4.3.3 Picture parameter set RBSP semantics

7.4.3.3.1 General picture parameter set RBSP semantics

pps_pic_parameter_set_id identifies the PPS for reference by other syntax elements. The value of `pps_pic_parameter_set_id` shall be in the range of 0 to 63, inclusive.

pps_seq_parameter_set_id specifies the value of `sps_seq_parameter_set_id` for the active SPS. The value of `pps_seq_parameter_set_id` shall be in the range of 0 to 15, inclusive.

dependent_slice_segments_enabled_flag equal to 1 specifies the presence of the syntax element `dependent_slice_segment_flag` in the slice segment headers for coded pictures referring to the PPS. `dependent_slice_segments_enabled_flag` equal to 0 specifies the absence of the syntax element `dependent_slice_segment_flag` in the slice segment headers for coded pictures referring to the PPS.

output_flag_present_flag equal to 1 indicates that the `pic_output_flag` syntax element is present in the associated slice headers. `output_flag_present_flag` equal to 0 indicates that the `pic_output_flag` syntax element is not present in the associated slice headers.

num_extra_slice_header_bits specifies the number of extra slice header bits that are present in the slice header RBSP for coded pictures referring to the PPS. The value of `num_extra_slice_header_bits` shall be in the range of 0 to 2, inclusive, in

bitstreams conforming to this version of this Specification. Other values for `num_extra_slice_header_bits` are reserved for future use by ITU-T | ISO/IEC. However, decoders shall allow `num_extra_slice_header_bits` to have any value.

sign_data_hiding_enabled_flag equal to 0 specifies that sign bit hiding is disabled. `sign_data_hiding_enabled_flag` equal to 1 specifies that sign bit hiding is enabled.

cabac_init_present_flag equal to 1 specifies that `cabac_init_flag` is present in slice headers referring to the PPS. `cabac_init_present_flag` equal to 0 specifies that `cabac_init_flag` is not present in slice headers referring to the PPS.

num_ref_idx_l0_default_active_minus1 specifies the inferred value of `num_ref_idx_l0_active_minus1` for P and B slices with `num_ref_idx_active_override_flag` equal to 0. The value of `num_ref_idx_l0_default_active_minus1` shall be in the range of 0 to 14, inclusive.

num_ref_idx_l1_default_active_minus1 specifies the inferred value of `num_ref_idx_l1_active_minus1` for B slices with `num_ref_idx_active_override_flag` equal to 0. The value of `num_ref_idx_l1_default_active_minus1` shall be in the range of 0 to 14, inclusive.

init_qp_minus26 plus 26 specifies the initial value of `SliceQpY` for each slice referring to the PPS. The initial value of `SliceQpY` is modified at the slice segment layer when a non-zero value of `slice_qp_delta` is decoded. The value of `init_qp_minus26` shall be in the range of $-(26 + \text{QpBdOffset}_Y)$ to +25, inclusive.

constrained_intra_pred_flag equal to 0 specifies that intra prediction allows usage of residual data and decoded samples of neighbouring coding blocks coded using either intra or inter prediction modes. `constrained_intra_pred_flag` equal to 1 specifies constrained intra prediction, in which case intra prediction only uses residual data and decoded samples from neighbouring coding blocks coded using intra prediction modes.

NOTE 1 – Encoders that operate in error-prone environments should be designed with consideration of the potential for error propagation caused by references to other pictures and references to areas within the current picture that use other pictures as references.

transform_skip_enabled_flag equal to 1 specifies that `transform_skip_flag` may be present in the residual coding syntax. `transform_skip_enabled_flag` equal to 0 specifies that `transform_skip_flag` is not present in the residual coding syntax.

cu_qp_delta_enabled_flag equal to 1 specifies that the `diff_cu_qp_delta_depth` syntax element is present in the PPS and that `cu_qp_delta_abs` may be present in the transform unit syntax and the palette syntax. `cu_qp_delta_enabled_flag` equal to 0 specifies that the `diff_cu_qp_delta_depth` syntax element is not present in the PPS and that `cu_qp_delta_abs` is not present in the transform unit syntax and the palette syntax.

diff_cu_qp_delta_depth specifies the difference between the luma CTB size and the minimum luma coding block size of coding units that convey `cu_qp_delta_abs` and `cu_qp_delta_sign_flag`. The value of `diff_cu_qp_delta_depth` shall be in the range of 0 to $\log_2 \text{diff_max_min_luma_coding_block_size}$, inclusive. When not present, the value of `diff_cu_qp_delta_depth` is inferred to be equal to 0.

The variable `Log2MinCuQpDeltaSize` is derived as follows:

$$\text{Log2MinCuQpDeltaSize} = \text{CtbLog2Size}_Y - \text{diff_cu_qp_delta_depth} \quad (7-36)$$

pps_cb_qp_offset and **pps_cr_qp_offset** specify the offsets to the luma quantization parameter `Qp'Y` used for deriving `Qp'cb` and `Qp'cr`, respectively. The values of `pps_cb_qp_offset` and `pps_cr_qp_offset` shall be in the range of -12 to +12, inclusive. When `ChromaArrayType` is equal to 0, `pps_cb_qp_offset` and `pps_cr_qp_offset` are not used in the decoding process and decoders shall ignore their value.

pps_slice_chroma_qp_offsets_present_flag equal to 1 indicates that the `slice_cb_qp_offset` and `slice_cr_qp_offset` syntax elements are present in the associated slice headers. `pps_slice_chroma_qp_offsets_present_flag` equal to 0 indicates that these syntax elements are not present in the associated slice headers. When `ChromaArrayType` is equal to 0, `pps_slice_chroma_qp_offsets_present_flag` shall be equal to 0.

weighted_pred_flag equal to 0 specifies that weighted prediction is not applied to P slices. `weighted_pred_flag` equal to 1 specifies that weighted prediction is applied to P slices.

weighted_bipred_flag equal to 0 specifies that the default weighted prediction is applied to B slices. `weighted_bipred_flag` equal to 1 specifies that weighted prediction is applied to B slices.

transquant_bypass_enabled_flag equal to 1 specifies that `cu_transquant_bypass_flag` is present. `transquant_bypass_enabled_flag` equal to 0 specifies that `cu_transquant_bypass_flag` is not present.

tiles_enabled_flag equal to 1 specifies that there is more than one tile in each picture referring to the PPS. `tiles_enabled_flag` equal to 0 specifies that there is only one tile in each picture referring to the PPS.

It is a requirement of bitstream conformance that the value of `tiles_enabled_flag` shall be the same for all PPSs that are activated within a CVS.

entropy_coding_sync_enabled_flag equal to 1 specifies that a specific synchronization process for context variables, and when applicable, Rice parameter initialization states and palette predictor variables, is invoked before decoding the CTU which includes the first CTB of a row of CTBs in each tile in each picture referring to the PPS, and a specific storage process for context variables, and when applicable, Rice parameter initialization states and palette predictor variables, is invoked after decoding the CTU which includes the second CTB of a row of CTBs in each tile in each picture referring to the PPS. **entropy_coding_sync_enabled_flag** equal to 0 specifies that no specific synchronization process for context variables, and when applicable, Rice parameter initialization states and palette predictor variables, is required to be invoked before decoding the CTU which includes the first CTB of a row of CTBs in each tile in each picture referring to the PPS, and no specific storage process for context variables, and when applicable, Rice parameter initialization states and palette predictor variables, is required to be invoked after decoding the CTU which includes the second CTB of a row of CTBs in each tile in each picture referring to the PPS.

It is a requirement of bitstream conformance that the value of **entropy_coding_sync_enabled_flag** shall be the same for all PPSs that are activated within a CVS.

When **entropy_coding_sync_enabled_flag** is equal to 1 and the first CTB in a slice is not the first CTB of a row of CTBs in a tile, it is a requirement of bitstream conformance that the last CTB in the slice shall belong to the same row of CTBs as the first CTB in the slice.

When **entropy_coding_sync_enabled_flag** is equal to 1 and the first CTB in a slice segment is not the first CTB of a row of CTBs in a tile, it is a requirement of bitstream conformance that the last CTB in the slice segment shall belong to the same row of CTBs as the first CTB in the slice segment.

num_tile_columns_minus1 plus 1 specifies the number of tile columns partitioning the picture. **num_tile_columns_minus1** shall be in the range of 0 to $\text{PicWidthInCtbsY} - 1$, inclusive. When not present, the value of **num_tile_columns_minus1** is inferred to be equal to 0.

num_tile_rows_minus1 plus 1 specifies the number of tile rows partitioning the picture. **num_tile_rows_minus1** shall be in the range of 0 to $\text{PicHeightInCtbsY} - 1$, inclusive. When not present, the value of **num_tile_rows_minus1** is inferred to be equal to 0.

When **tiles_enabled_flag** is equal to 1, **num_tile_columns_minus1** and **num_tile_rows_minus1** shall not be both equal to 0.

uniform_spacing_flag equal to 1 specifies that tile column boundaries and likewise tile row boundaries are distributed uniformly across the picture. **uniform_spacing_flag** equal to 0 specifies that tile column boundaries and likewise tile row boundaries are not distributed uniformly across the picture but signalled explicitly using the syntax elements **column_width_minus1[i]** and **row_height_minus1[i]**. When not present, the value of **uniform_spacing_flag** is inferred to be equal to 1.

column_width_minus1[i] plus 1 specifies the width of the i-th tile column in units of CTBs.

row_height_minus1[i] plus 1 specifies the height of the i-th tile row in units of CTBs.

The following variables are derived by invoking the CTB raster and tile scanning conversion process as specified in clause 6.5.1:

- The list **CtbAddrRsToTs[ctbAddrRs]** for **ctbAddrRs** ranging from 0 to $\text{PicSizeInCtbsY} - 1$, inclusive, specifying the conversion from a CTB address in the CTB raster scan of a picture to a CTB address in the tile scan,
- the list **CtbAddrTsToRs[ctbAddrTs]** for **ctbAddrTs** ranging from 0 to $\text{PicSizeInCtbsY} - 1$, inclusive, specifying the conversion from a CTB address in the tile scan to a CTB address in the CTB raster scan of a picture,
- the list **TileId[ctbAddrTs]** for **ctbAddrTs** ranging from 0 to $\text{PicSizeInCtbsY} - 1$, inclusive, specifying the conversion from a CTB address in tile scan to a tile ID,
- the list **ColumnWidthInLumaSamples[i]** for **i** ranging from 0 to **num_tile_columns_minus1**, inclusive, specifying the width of the i-th tile column in units of luma samples,
- the list **RowHeightInLumaSamples[j]** for **j** ranging from 0 to **num_tile_rows_minus1**, inclusive, specifying the height of the j-th tile row in units of luma samples.

The values of **ColumnWidthInLumaSamples[i]** for **i** ranging from 0 to **num_tile_columns_minus1**, inclusive, and **RowHeightInLumaSamples[j]** for **j** ranging from 0 to **num_tile_rows_minus1**, inclusive, shall all be greater than 0.

The array **MinTbAddrZs** with elements **MinTbAddrZs[x][y]** for **x** ranging from 0 to $(\text{PicWidthInCtbsY} \ll (\text{CtbLog2SizeY} - \text{MinTbLog2SizeY})) - 1$, inclusive, and **y** ranging from 0 to $(\text{PicHeightInCtbsY} \ll (\text{CtbLog2SizeY} - \text{MinTbLog2SizeY})) - 1$, inclusive, specifying the conversion from a location (x, y) in units of minimum transform blocks to a transform block address in z-scan order, is derived by invoking the z-scan order array initialization process as specified in clause 6.5.2.

loop_filter_across_tiles_enabled_flag equal to 1 specifies that in-loop filtering operations may be performed across tile boundaries in pictures referring to the PPS. **loop_filter_across_tiles_enabled_flag** equal to 0 specifies that in-loop filtering

operations are not performed across tile boundaries in pictures referring to the PPS. The in-loop filtering operations include the deblocking filter and sample adaptive offset filter operations. When not present, the value of `loop_filter_across_tiles_enabled_flag` is inferred to be equal to 1.

pps_loop_filter_across_slices_enabled_flag equal to 1 specifies that in-loop filtering operations may be performed across left and upper boundaries of slices referring to the PPS. `pps_loop_filter_across_slices_enabled_flag` equal to 0 specifies that in-loop filtering operations are not performed across left and upper boundaries of slices referring to the PPS. The in-loop filtering operations include the deblocking filter and sample adaptive offset filter operations.

NOTE 2 – Loop filtering across slice boundaries can be enabled while loop filtering across tile boundaries is disabled and vice versa.

deblocking_filter_control_present_flag equal to 1 specifies the presence of deblocking filter control syntax elements in the PPS. `deblocking_filter_control_present_flag` equal to 0 specifies the absence of deblocking filter control syntax elements in the PPS.

deblocking_filter_override_enabled_flag equal to 1 specifies the presence of `deblocking_filter_override_flag` in the slice headers for pictures referring to the PPS. `deblocking_filter_override_enabled_flag` equal to 0 specifies the absence of `deblocking_filter_override_flag` in the slice headers for pictures referring to the PPS. When not present, the value of `deblocking_filter_override_enabled_flag` is inferred to be equal to 0.

pps_deblocking_filter_disabled_flag equal to 1 specifies that the operation of deblocking filter is not applied for slices referring to the PPS in which `slice_deblocking_filter_disabled_flag` is not present. `pps_deblocking_filter_disabled_flag` equal to 0 specifies that the operation of the deblocking filter is applied for slices referring to the PPS in which `slice_deblocking_filter_disabled_flag` is not present. When not present, the value of `pps_deblocking_filter_disabled_flag` is inferred to be equal to 0.

pps_beta_offset_div2 and **pps_tc_offset_div2** specify the default deblocking parameter offsets for β and tC (divided by 2) that are applied for slices referring to the PPS, unless the default deblocking parameter offsets are overridden by the deblocking parameter offsets present in the slice headers of the slices referring to the PPS. The values of `pps_beta_offset_div2` and `pps_tc_offset_div2` shall both be in the range of -6 to 6 , inclusive. When not present, the value of `pps_beta_offset_div2` and `pps_tc_offset_div2` are inferred to be equal to 0.

pps_scaling_list_data_present_flag equal to 1 specifies that the scaling list data used for the pictures referring to the PPS are derived based on the scaling lists specified by the active SPS and the scaling lists specified by the PPS. `pps_scaling_list_data_present_flag` equal to 0 specifies that the scaling list data used for the pictures referring to the PPS are inferred to be equal to those specified by the active SPS. When `scaling_list_enabled_flag` is equal to 0, the value of `pps_scaling_list_data_present_flag` shall be equal to 0. When `scaling_list_enabled_flag` is equal to 1, `sps_scaling_list_data_present_flag` is equal to 0 and `pps_scaling_list_data_present_flag` is equal to 0, the default scaling list data are used to derive the array `ScalingFactor` as described in the scaling list data semantics as specified in clause 7.4.5.

lists_modification_present_flag equal to 1 specifies that the syntax structure `ref_pic_lists_modification()` is present in the slice segment header. `lists_modification_present_flag` equal to 0 specifies that the syntax structure `ref_pic_lists_modification()` is not present in the slice segment header.

log2_parallel_merge_level_minus2 plus 2 specifies the value of the variable `Log2ParMrgLevel`, which is used in the derivation process for luma motion vectors for merge mode as specified in clause 8.5.3.2.2 and the derivation process for spatial merging candidates as specified in clause 8.5.3.2.3. The value of `log2_parallel_merge_level_minus2` shall be in the range of 0 to $CtbLog2SizeY - 2$, inclusive.

The variable `Log2ParMrgLevel` is derived as follows:

$$\text{Log2ParMrgLevel} = \text{log2_parallel_merge_level_minus2} + 2 \quad (7-37)$$

NOTE 3 – The value of `Log2ParMrgLevel` indicates the built-in capability of parallel derivation of the merging candidate lists. For example, when `Log2ParMrgLevel` is equal to 6, the merging candidate lists for all the prediction units (PUs) and coding units (CUs) contained in a 64×64 block can be derived in parallel.

slice_segment_header_extension_present_flag equal to 0 specifies that no slice segment header extension syntax elements are present in the slice segment headers for coded pictures referring to the PPS. `slice_segment_header_extension_present_flag` equal to 1 specifies that slice segment header extension syntax elements are present in the slice segment headers for coded pictures referring to the PPS.

pps_extension_present_flag equal to 1 specifies that the syntax elements `pps_range_extension_flag`, `pps_multilayer_extension_flag`, `pps_3d_extension_flag`, `pps_scc_extension_flag`, and `pps_extension_4bits` are present in the picture parameter set RBSP syntax structure. `pps_extension_present_flag` equal to 0 specifies that these syntax elements are not present.

pps_range_extension_flag equal to 1 specifies that the `pps_range_extension()` syntax structure is present in the PPS RBSP syntax structure. `pps_range_extension_flag` equal to 0 specifies that this syntax structure is not present. When not present, the value of `pps_range_extension_flag` is inferred to be equal to 0.

pps_multilayer_extension_flag equal to 1 specifies that the `pps_multilayer_extension()` syntax structure is present in the PPS RBSP syntax structure. `pps_multilayer_extension_flag` equal to 0 specifies that the `pps_multilayer_extension()` syntax structure is not present. When not present, the value of `pps_multilayer_extension_flag` is inferred to be equal to 0.

pps_3d_extension_flag equal to 1 specifies that the `pps_3d_extension()` syntax structure (specified in Annex I) is present in the PPS RBSP syntax structure. `pps_3d_extension_flag` equal to 0 specifies that the `pps_3d_extension()` syntax structure is not present. When not present, the value of `pps_3d_extension_flag` is inferred to be equal to 0.

pps_scc_extension_flag equal to 1 specifies that the `pps_scc_extension()` syntax structure is present in the PPS RBSP syntax structure. `pps_scc_extension_flag` equal to 0 specifies that this syntax structure is not present. When not present, the value of `pps_scc_extension_flag` is inferred to be equal to 0.

pps_extension_4bits equal to 0 specifies that no `pps_extension_data_flag` syntax elements are present in the PPS RBSP syntax structure. When present, `pps_extension_4bits` shall be equal to 0 in bitstreams conforming to this version of this Specification. Values of `pps_extension_4bits` not equal to 0 are reserved for future use by ITU-T | ISO/IEC. Decoders shall allow the value of `pps_extension_4bits` to be not equal to 0 and shall ignore all `pps_extension_data_flag` syntax elements in a PPS NAL unit. When not present, the value of `pps_extension_4bits` is inferred to be equal to 0.

pps_extension_data_flag may have any value. Its presence and value do not affect the decoding process specified in this version of this Specification. Decoders conforming to this version of this Specification shall ignore all `pps_extension_data_flag` syntax elements.

7.4.3.3.2 Picture parameter set range extension semantics

log2_max_transform_skip_block_size_minus2 plus 2 specifies the maximum transform block size for which `transform_skip_flag` may be present in coded pictures referring to the PPS. When not present, the value of `log2_max_transform_skip_block_size_minus2` is inferred to be equal to 0. When present, the value of `log2_max_transform_skip_block_size_minus2` shall be less than or equal to `MaxTbLog2SizeY - 2`.

The variable `Log2MaxTransformSkipSize` is derived as follows:

$$\text{Log2MaxTransformSkipSize} = \text{log2_max_transform_skip_block_size_minus2} + 2 \quad (7-38)$$

cross_component_prediction_enabled_flag equal to 1 specifies that `log2_res_scale_abs_plus1` and `res_scale_sign_flag` may be present in the transform unit syntax for pictures referring to the PPS. `cross_component_prediction_enabled_flag` equal to 0 specifies that `log2_res_scale_abs_plus1` and `res_scale_sign_flag` are not present for pictures referring to the PPS. When not present, the value of `cross_component_prediction_enabled_flag` is inferred to be equal to 0. When `ChromaArrayType` is not equal to 3, it is a requirement of bitstream conformance that the value of `cross_component_prediction_enabled_flag` shall be equal to 0.

chroma_qp_offset_list_enabled_flag equal to 1 specifies that the `cu_chroma_qp_offset_flag` may be present in the transform unit syntax. `chroma_qp_offset_list_enabled_flag` equal to 0 specifies that the `cu_chroma_qp_offset_flag` is not present in the transform unit syntax. When `ChromaArrayType` is equal to 0, it is a requirement of bitstream conformance that the value of `chroma_qp_offset_list_enabled_flag` shall be equal to 0.

diff_cu_chroma_qp_offset_depth specifies the difference between the luma CTB size and the minimum luma coding block size of coding units that convey `cu_chroma_qp_offset_flag`. The value of `diff_cu_chroma_qp_offset_depth` shall be in the range of 0 to `log2_diff_max_min_luma_coding_block_size`, inclusive.

The variable `Log2MinCuChromaQpOffsetSize` is derived as follows:

$$\text{Log2MinCuChromaQpOffsetSize} = \text{CtbLog2SizeY} - \text{diff_cu_chroma_qp_offset_depth} \quad (7-39)$$

chroma_qp_offset_list_len_minus1 plus 1 specifies the number of `cb_qp_offset_list[i]` and `cr_qp_offset_list[i]` syntax elements that are present in the PPS. The value of `chroma_qp_offset_list_len_minus1` shall be in the range of 0 to 5, inclusive.

cb_qp_offset_list[i] and **cr_qp_offset_list[i]** specify offsets used in the derivation of Qp'_{Cb} and Qp'_{Cr} , respectively. The values of `cb_qp_offset_list[i]` and `cr_qp_offset_list[i]` shall be in the range of -12 to +12, inclusive.

log2_sao_offset_scale_luma is the base 2 logarithm of the scaling parameter that is used to scale sample adaptive offset (SAO) offset values for luma samples. The value of `log2_sao_offset_scale_luma` shall be in the range of 0 to $\text{Max}(0, \text{BitDepth}_Y - 10)$, inclusive. When not present, the value of `log2_sao_offset_scale_luma` is inferred to be equal to 0.

log2_sao_offset_scale_chroma is the base 2 logarithm of the scaling parameter that is used to scale SAO offset values for chroma samples. The value of log2_sao_offset_scale_chroma shall be in the range of 0 to Max(0, BitDepth_C – 10), inclusive. When not present, the value of log2_sao_offset_scale_chroma is inferred to be equal to 0.

7.4.3.3.3 Picture parameter set screen content coding extension semantics

pps_curr_pic_ref_enabled_flag equal to 1 specifies that a picture referring to the PPS may be included in a reference picture list of a slice of the picture itself. pps_curr_pic_ref_enabled_flag equal to 0 specifies that a picture referring to the PPS is never included in a reference picture list of a slice of the picture itself. When not present, the value of pps_curr_pic_ref_enabled_flag is inferred to be equal to 0.

It is a requirement of bitstream conformance that when sps_curr_pic_ref_enabled_flag is equal to 0, the value of pps_curr_pic_ref_enabled_flag shall be equal to 0.

The variable TwoVersionsOfCurrDecPicFlag is derived as follows:

$$\text{TwoVersionsOfCurrDecPicFlag} = \text{pps_curr_pic_ref_enabled_flag} \ \&\& \\ (\text{sample_adaptive_offset_enabled_flag} \ || \ !\text{pps_deblocking_filter_disabled_flag} \ || \ \text{deblocking_filter_override_enabled_flag}) \quad (7-40)$$

When sps_max_dec_pic_buffering_minus1[TemporalId] is equal to 0, the value of TwoVersionsOfCurrDecPicFlag shall be equal to 0.

residual_adaptive_colour_transform_enabled_flag equal to 1 specifies that an adaptive colour transform may be applied to the residual in the decoding process. residual_adaptive_colour_transform_enabled_flag equal to 0 specifies that adaptive colour transform is not applied to the residual. When not present, the value of residual_adaptive_colour_transform_enabled_flag is inferred to be equal to 0.

When ChromaArrayType is not equal to 3, residual_adaptive_colour_transform_enabled_flag shall be equal to 0.

pps_slice_act_qp_offsets_present_flag equal to 1 specifies that slice_act_y_qp_offset, slice_act_cb_qp_offset, slice_act_cr_qp_offset are present in the slice header. pps_slice_act_qp_offsets_present_flag equal to 0 specifies that slice_act_y_qp_offset, slice_act_cb_qp_offset, slice_act_cr_qp_offset are not present in the slice header. When not present, the value of pps_slice_act_qp_offsets_present_flag is inferred to be equal to 0.

pps_act_y_qp_offset_plus5, **pps_act_cb_qp_offset_plus5** and **pps_act_cr_qp_offset_plus3** are used to determine the offsets that are applied to the quantization parameter values qP derived in clause 8.6.2 for the luma, Cb and Cr components, respectively, when tu_residual_act_flag[xTbY][yTbY] is equal to 1. When not present, the values of pps_act_y_qp_offset_plus5, pps_act_cb_qp_offset_plus5 and pps_act_cr_qp_offset_plus3 are inferred to be equal to 0.

The variable PpsActQpOffsetY is set equal to pps_act_y_qp_offset_plus5 – 5.

The variable PpsActQpOffsetCb is set equal to pps_act_cb_qp_offset_plus5 – 5.

The variable PpsActQpOffsetCr is set equal to pps_act_cr_qp_offset_plus3 – 3.

NOTE – The constant offset values of 5, 5, and 3 above are applied because the adaptive colour transformation that is applied to the residual when tu_residual_act_flag[xTbY][yTbY] is equal to 1 is not an orthonormal transformation.

It is a requirement of bitstream conformance that the values of PpsActQpOffsetY, PpsActQpOffsetCb, and PpsActQpOffsetCr shall be in the range of –12 to +12, inclusive.

pps_palette_predictor_initializers_present_flag equal to 1 specifies that the palette predictor initializers used for the pictures referring to the PPS are derived based on the palette predictor initializers specified by the PPS. pps_palette_predictor_initializer_flag equal to 0 specifies that the palette predictor initializers used for the pictures referring to the PPS are inferred to be equal to those specified by the active SPS. When not present, the value of pps_palette_predictor_initializers_present_flag is inferred to be equal to 0.

It is a requirement of bitstream conformance that the value of pps_palette_predictor_initializers_present_flag shall be equal to 0 when either palette_max_size is equal to 0 or palette_mode_enabled_flag is equal to 0.

pps_num_palette_predictor_initializers specifies the number of entries in the picture palette predictor initializer.

It is a requirement of bitstream conformance that the value of pps_num_palette_predictor_initializers shall be less than or equal to PaletteMaxPredictorSize.

monochrome_palette_flag equal to 1 specifies that the pictures that refer to this PPS are monochrome. monochrome_palette_flag equal to 0 specifies that the pictures that refer to this PPS have multiple components.

It is a requirement of bitstream conformance that the value of the monochrome_palette_flag shall be equal to (chroma_format_idc == 0).

luma_bit_depth_entry_minus8 plus 8 specifies the bit depth of the luma component of the entries of the palette predictor initializer. It is a requirement of bitstream conformance that the value of **luma_bit_depth_entry_minus8** shall be equal to the value of **bit_depth_luma_minus8**.

chroma_bit_depth_entry_minus8 plus 8 specifies the bit depth of the chroma components of the entries of the palette predictor initializer. It is a requirement of bitstream conformance that the value of **chroma_bit_depth_entry_minus8** shall be equal to the value of **bit_depth_chroma_minus8**.

pps_palette_predictor_initializer[comp][i] specifies the value of the comp-th component of the i-th palette entry in the PPS that is used to initialize the array **PredictorPaletteEntries**. For values of i in the range of 0 to **pps_num_palette_predictor_initializers** – 1, inclusive, the number of bits used to represent **pps_palette_predictor_initializer**[0][i] is **luma_bit_depth_entry_minus8** + 8, and the number of bits used to represent **pps_palette_predictor_initializer**[1][i] and **pps_palette_predictor_initializer**[2][i] is **chroma_bit_depth_entry_minus8** + 8.

7.4.3.4 Supplemental enhancement information RBSP semantics

Supplemental enhancement information (SEI) contains information that is not necessary to decode the samples of coded pictures from VCL NAL units. An SEI RBSP contains one or more SEI messages.

7.4.3.5 Access unit delimiter RBSP semantics

The access unit delimiter may be used to indicate the type of slices present in the coded pictures in the access unit containing the access unit delimiter NAL unit and to simplify the detection of the boundary between access units. There is no normative decoding process associated with the access unit delimiter.

pic_type indicates that the slice_type values for all slices of the coded pictures in the access unit containing the access unit delimiter NAL unit are members of the set listed in Table 7-2 for the given value of **pic_type**. The value of **pic_type** shall be equal to 0, 1 or 2 in bitstreams conforming to this version of this Specification. Other values of **pic_type** are reserved for future use by ITU-T | ISO/IEC. Decoders conforming to this version of this Specification shall ignore reserved values of **pic_type**.

Table 7-2 – Interpretation of **pic_type**

pic_type	slice_type values that may be present in the coded picture
0	I
1	P, I
2	B, P, I

7.4.3.6 End of sequence RBSP semantics

When included in a NAL unit with **nuh_layer_id** equal to 0, the end of sequence RBSP specifies that the current access unit is the last access unit in the coded video sequence in decoding order and the next subsequent access unit in the bitstream in decoding order (if any) is an IRAP access unit with **NoRaslOutputFlag** equal to 1. The syntax content of the SODB and RBSP for the end of sequence RBSP are empty.

7.4.3.7 End of bitstream RBSP semantics

The end of bitstream RBSP indicates that no additional NAL units are present in the bitstream that are subsequent to the end of bitstream RBSP in decoding order. The syntax content of the SODB and RBSP for the end of bitstream RBSP are empty.

NOTE – When an elementary stream contains more than one bitstream, the last NAL unit of the last access unit of a bitstream must contain an end of bitstream NAL unit and the first access unit of the subsequent bitstream must be an IRAP access unit. This IRAP access unit may be a CRA, BLA or IDR access unit.

7.4.3.8 Filler data RBSP semantics

The filler data RBSP contains bytes whose value shall be equal to 0xFF. No normative decoding process is specified for a filler data RBSP.

ff_byte is a byte equal to 0xFF.

7.4.3.9 Slice segment layer RBSP semantics

The slice segment layer RBSP consists of a slice segment header and slice segment data.

7.4.3.10 RBSP slice segment trailing bits semantics

cabac_zero_word is a byte-aligned sequence of two bytes equal to 0x0000.

Let **NumBytesInVclNalUnits** be the sum of the values of **NumBytesInNalUnit** for all VCL NAL units of a coded picture.

Let **BinCountsInNalUnits** be the number of times that the parsing process function **DecodeBin()**, specified in clause 9.3.4.3, is invoked to decode the contents of all VCL NAL units of a coded picture.

Let the variable **RawMinCuBits** be derived as follows:

$$\text{RawMinCuBits} = \text{MinCbSizeY} * \text{MinCbSizeY} * (\text{BitDepth}_Y + 2 * \text{BitDepth}_C / (\text{SubWidthC} * \text{SubHeightC})) \quad (7-41)$$

The value of **BinCountsInNalUnits** shall be less than or equal to $(32 \div 3) * \text{NumBytesInVclNalUnits} + (\text{RawMinCuBits} * \text{PicSizeInMinCbsY}) \div 32$.

NOTE – The constraint on the maximum number of bins resulting from decoding the contents of the coded slice segment NAL units can be met by inserting a number of **cabac_zero_word** syntax elements to increase the value of **NumBytesInVclNalUnits**. Each **cabac_zero_word** is represented in a NAL unit by the three-byte sequence 0x000003 (as a result of the constraints on NAL unit contents that result in requiring inclusion of an **emulation_prevention_three_byte** for each **cabac_zero_word**).

7.4.3.11 RBSP trailing bits semantics

rbstop_one_bit shall be equal to 1.

rbstop_alignment_zero_bit shall be equal to 0.

7.4.3.12 Byte alignment semantics

alignment_bit_equal_to_one shall be equal to 1.

alignment_bit_equal_to_zero shall be equal to 0.

7.4.4 Profile, tier and level semantics

general_profile_space specifies the context for the interpretation of **general_profile_idc** and **general_profile_compatibility_flag[j]** for all values of **j** in the range of 0 to 31, inclusive. The value of **general_profile_space** shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for **general_profile_space** are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the CVS when **general_profile_space** is not equal to 0.

general_tier_flag specifies the tier context for the interpretation of **general_level_idc** as specified in Annex A.

general_profile_idc, when **general_profile_space** is equal to 0, indicates a profile to which the CVS conforms as specified in Annex A. Bitstreams shall not contain values of **general_profile_idc** other than those specified in Annex A. Other values of **general_profile_idc** are reserved for future use by ITU-T | ISO/IEC.

general_profile_compatibility_flag[j] equal to 1, when **general_profile_space** is equal to 0, indicates that the CVS conforms to the profile indicated by **general_profile_idc** equal to **j** as specified in Annex A. When **general_profile_space** is equal to 0, **general_profile_compatibility_flag[general_profile_idc]** shall be equal to 1. The value of **general_profile_compatibility_flag[j]** shall be equal to 0 for any value of **j** that is not specified as an allowed value of **general_profile_idc** in Annex A.

general_progressive_source_flag and **general_interlaced_source_flag** are interpreted as follows:

- If **general_progressive_source_flag** is equal to 1 and **general_interlaced_source_flag** is equal to 0, the source scan type of the pictures in the CVS should be interpreted as progressive only.
- Otherwise, if **general_progressive_source_flag** is equal to 0 and **general_interlaced_source_flag** is equal to 1, the source scan type of the pictures in the CVS should be interpreted as interlaced only.
- Otherwise, if **general_progressive_source_flag** is equal to 0 and **general_interlaced_source_flag** is equal to 0, the source scan type of the pictures in the CVS should be interpreted as unknown or unspecified.
- Otherwise (**general_progressive_source_flag** is equal to 1 and **general_interlaced_source_flag** is equal to 1), the source scan type of each picture in the CVS is indicated at the picture level using the syntax element **source_scan_type** in a picture timing SEI message.

NOTE 1 – Decoders may ignore the values of **general_progressive_source_flag** and **general_interlaced_source_flag** for purposes other than determining the value to be inferred for **frame_field_info_present_flag** when **vui_parameters_present_flag** is equal to 0, as there are no other decoding process requirements associated with the values of these flags. Moreover, the actual source scan type of the pictures is outside the scope of this Specification and the method by which the encoder selects the values of **general_progressive_source_flag** and **general_interlaced_source_flag** is unspecified.

general_non_packed_constraint_flag equal to 1 specifies that there are no frame packing arrangement SEI messages, segmented rectangular frame packing arrangement SEI messages, equirectangular projection SEI messages, or cubemap projection SEI messages present in the CVS. **general_non_packed_constraint_flag** equal to 0 indicates that there may or may not be one or more frame packing arrangement SEI messages, segmented rectangular frame packing arrangement SEI messages, equirectangular projection SEI messages, or cubemap projection SEI messages present in the CVS.

NOTE 2 – Decoders may ignore the value of **general_non_packed_constraint_flag**, as there are no decoding process requirements associated with the presence or interpretation of frame packing arrangement SEI messages, segmented rectangular frame packing arrangement SEI messages, equirectangular projection SEI messages, or cubemap projection SEI messages.

general_frame_only_constraint_flag equal to 1 specifies that **field_seq_flag** is equal to 0. **general_frame_only_constraint_flag** equal to 0 indicates that **field_seq_flag** may or may not be equal to 0.

NOTE 3 – Decoders may ignore the value of **general_frame_only_constraint_flag**, as there are no decoding process requirements associated with the value of **field_seq_flag**.

NOTE 4 – When **general_progressive_source_flag** is equal to 1, **general_frame_only_constraint_flag** may or may not be equal to 1.

general_max_12bit_constraint_flag, **general_max_10bit_constraint_flag**, **general_max_8bit_constraint_flag**, **general_max_422chroma_constraint_flag**, **general_max_420chroma_constraint_flag**, **general_max_monochrome_constraint_flag**, **general_intra_constraint_flag**, **general_one_picture_only_constraint_flag**, **general_lower_bit_rate_constraint_flag**, and **general_max_14bit_constraint_flag**, when present, have semantics specified in Annex A when the profile indicated by **general_profile_idc** or **general_profile_compatibility_flag[j]** is a profile specified in Annex A.

general_reserved_zero_33bits, when present, shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for **general_reserved_zero_33bits** are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of **general_reserved_zero_33bits**.

general_reserved_zero_34bits, when present, shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for **general_reserved_zero_34bits** are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of **general_reserved_zero_34bits**.

general_reserved_zero_7bits, when present, shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for **general_reserved_zero_7bits** are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of **general_reserved_zero_7bits**.

general_reserved_zero_35bits, when present, shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for **general_reserved_zero_35bits** are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of **general_reserved_zero_35bits**.

general_reserved_zero_43bits, when present, shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for **general_reserved_zero_43bits** are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of **general_reserved_zero_43bits**.

general_inbld_flag equal to 1 specifies that the INBLD capability as specified in Annex F is required for decoding of the layer to which the **profile_tier_level()** syntax structure applies. **general_inbld_flag** equal to 0 specifies that the INBLD capability as specified in Annex F is not required for decoding of the layer to which the **profile_tier_level()** syntax structure applies. When **profilePresentFlag** is equal to 1, **general_profile_idc** is not equal to 9 or 11 and is not in the range of 1 to 5, inclusive, **general_profile_compatibility_flag[9]** is not equal to 1, **general_profile_compatibility_flag[11]** is not equal to 1, and **general_profile_compatibility_flag[j]** is not equal to 1 for any value of **j** in the range of 1 to 5, inclusive, the value of **general_inbld_flag** is inferred to be equal to 0.

general_reserved_zero_bit, when present, shall be equal to 0 in bitstreams conforming to this version of this Specification. The value 1 for **general_reserved_zero_1bit** is reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of **general_reserved_zero_bit**.

general_level_idc indicates a level to which the CVS conforms as specified in Annex A. Bitstreams shall not contain values of **general_level_idc** other than those specified in Annex A. Other values of **general_level_idc** are reserved for future use by ITU-T | ISO/IEC.

NOTE 5 – A greater value of **general_level_idc** indicates a higher level. The maximum level signalled in the VPS for a CVS may be higher than the level signalled in the SPS for the same CVS.

NOTE 6 – When the coded video sequence conforms to multiple profiles, **general_profile_idc** should indicate the profile that provides the preferred decoded result or the preferred bitstream identification, as determined by the encoder (in a manner not specified in this Specification).

NOTE 7 – The syntax elements **general_reserved_zero_33bits**, **general_reserved_zero_34bits** and **general_reserved_zero_43bits** may be used in future versions of this Specification to indicate further constraints on the bitstream (e.g., that a particular syntax combination that would otherwise be permitted by the indicated values of **general_profile_compatibility_flag[j]**, is not used).

sub_layer_profile_present_flag[i] equal to 1, specifies that profile information is present in the **profile_tier_level()** syntax structure for the sub-layer representation with **TemporalId** equal to **i**. **sub_layer_profile_present_flag[i]** equal to 0

specifies that profile information is not present in the `profile_tier_level()` syntax structure for the sub-layer representation with `TemporalId` equal to `i`. When `profilePresentFlag` is equal to 0, `sub_layer_profile_present_flag[i]` shall be equal to 0.

`sub_layer_level_present_flag[i]` equal to 1 specifies that level information is present in the `profile_tier_level()` syntax structure for the sub-layer representation with `TemporalId` equal to `i`. `sub_layer_level_present_flag[i]` equal to 0 specifies that level information is not present in the `profile_tier_level()` syntax structure for the sub-layer representation with `TemporalId` equal to `i`.

`reserved_zero_2bits[i]` shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for `reserved_zero_2bits[i]` are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of `reserved_zero_2bits[i]`.

Each of the following syntax elements:

- `sub_layer_profile_space[i]`,
- `sub_layer_tier_flag[i]`,
- `sub_layer_profile_idc[i]`,
- `sub_layer_profile_compatibility_flag[i][j]`,
- `sub_layer_progressive_source_flag[i]`,
- `sub_layer_interlaced_source_flag[i]`,
- `sub_layer_non_packed_constraint_flag[i]`,
- `sub_layer_frame_only_constraint_flag[i]`,
- `sub_layer_max_12bit_constraint_flag[i]`,
- `sub_layer_max_10bit_constraint_flag[i]`,
- `sub_layer_max_8bit_constraint_flag[i]`,
- `sub_layer_max_422chroma_constraint_flag[i]`,
- `sub_layer_max_420chroma_constraint_flag[i]`,
- `sub_layer_max_monochrome_constraint_flag[i]`,
- `sub_layer_intra_constraint_flag[i]`,
- `sub_layer_one_picture_only_constraint_flag[i]`,
- `sub_layer_lower_bit_rate_constraint_flag[i]`,
- `sub_layer_max_14bit_constraint_flag[i]`,
- `sub_layer_reserved_zero_33bits[i]`,
- `sub_layer_reserved_zero_34bits[i]`,
- `sub_layer_reserved_zero_7bits[i]`,
- `sub_layer_reserved_zero_35bits[i]`,
- `sub_layer_reserved_zero_43bits[i]`,
- `sub_layer_inbld_flag[i]`,
- `sub_layer_reserved_zero_bit[i]`, and
- `sub_layer_level_idc[i]`,

is referred to as the `i`-th corresponding sub-layer syntax element of each of the following syntax elements:

- `general_profile_space`,
- `general_tier_flag`,
- `general_profile_idc`,
- `general_profile_compatibility_flag[j]`,
- `general_progressive_source_flag`,
- `general_interlaced_source_flag`,
- `general_non_packed_constraint_flag`,
- `general_frame_only_constraint_flag`,
- `general_max_12bit_constraint_flag`,
- `general_max_10bit_constraint_flag`,

- `general_max_8bit_constraint_flag`,
- `general_max_422chroma_constraint_flag`,
- `general_max_420chroma_constraint_flag`,
- `general_max_monochrome_constraint_flag`,
- `general_intra_constraint_flag`,
- `general_one_picture_only_constraint_flag`,
- `general_lower_bit_rate_constraint_flag`,
- `general_max_14bit_constraint_flag`,
- `general_reserved_zero_33bits`,
- `general_reserved_zero_34bits`,
- `general_reserved_zero_7bits`,
- `general_reserved_zero_35bits`,
- `general_reserved_zero_43bits`,
- `general_inbld_flag`,
- `general_reserved_zero_bit`, and
- `general_level_idc`,

respectively.

The semantics of a particular syntax element's i -th corresponding sub-layer syntax element, apart from the specification of the inference of the value when not present, is the same as for the particular syntax element, but applies to the sub-layer representation with `TemporalId` equal to i .

When not present, the value of `sub_layer_tier_flag[i]` is inferred to be equal to 0.

NOTE 8 – It is possible that `sub_layer_tier_flag[i]` is not present and `sub_layer_level_idc[i]` is present. In this case, a default value of `sub_layer_tier_flag[i]` is needed for interpretation of `sub_layer_level_idc[i]`.

When the `profile_tier_level()` syntax structure is included in an SPS or is the first `profile_tier_level()` syntax structure in a VPS, and any of the syntax elements `sub_layer_profile_space[i]`, `sub_layer_profile_idc[i]`, `sub_layer_profile_compatibility_flag[i][j]`, `sub_layer_progressive_source_flag[i]`, `sub_layer_interlaced_source_flag[i]`, `sub_layer_non_packed_constraint_flag[i]`, `sub_layer_frame_only_constraint_flag[i]`, `sub_layer_max_12bit_constraint_flag[i]`, `sub_layer_max_10bit_constraint_flag[i]`, `sub_layer_max_8bit_constraint_flag[i]`, `sub_layer_max_422chroma_constraint_flag[i]`, `sub_layer_max_420chroma_constraint_flag[i]`, `sub_layer_max_monochrome_constraint_flag[i]`, `sub_layer_intra_constraint_flag[i]`, `sub_layer_one_picture_only_constraint_flag[i]`, `sub_layer_lower_bit_rate_constraint_flag[i]`, `sub_layer_max_14bit_constraint_flag`, `sub_layer_reserved_zero_33bits[i]`, `sub_layer_reserved_zero_34bits[i]`, `sub_layer_reserved_zero_43bits[i]`, `sub_layer_inbld_flag[i]`, `sub_layer_reserved_zero_1bit[i]` and `sub_layer_level_idc[i]` is not present for any value of i in the range of 0 to `maxNumSubLayersMinus1 – 1`, inclusive, in the `profile_tier_level()` syntax structure, the value of the syntax element is inferred as follows (in decreasing order of i values from `maxNumSubLayersMinus1 – 1` to 0):

- If the value of i is equal to `maxNumSubLayersMinus1`, the value of the syntax element is inferred to be equal to the value of the corresponding syntax element prefixed with "general_" of the same `profile_tier_level()` syntax structure.

NOTE 9 – For example, in this case, if `sub_layer_profile_space[i]` is not present, the value is inferred to be equal to `general_profile_space` of the same `profile_tier_level()` syntax structure.

- Otherwise (the value of i is less than `maxNumSubLayersMinus1`), the value of the syntax element is inferred to be equal to the corresponding syntax element with i being replaced with $i + 1$ of the same `profile_tier_level()` syntax structure.

NOTE 10 – For example, in this case, if `sub_layer_profile_space[i]` is not present, the value is inferred to be equal to `sub_layer_profile_space[i + 1]` of the same `profile_tier_level()` syntax structure.

7.4.5 Scaling list data semantics

`scaling_list_pred_mode_flag[sizeId][matrixId]` equal to 0 specifies that the values of the scaling list are the same as the values of a reference scaling list. The reference scaling list is specified by `scaling_list_pred_matrix_id_delta[sizeId][matrixId]`. `scaling_list_pred_mode_flag[sizeId][matrixId]` equal to 1 specifies that the values of the scaling list are explicitly signalled.

`scaling_list_pred_matrix_id_delta[sizeId][matrixId]` specifies the reference scaling list used to derive `ScalingList[sizeId][matrixId]` as follows:

- If `scaling_list_pred_matrix_id_delta[sizeId][matrixId]` is equal to 0, the scaling list is inferred from the default scaling list `ScalingList[sizeId][matrixId][i]` as specified in Table 7-5 and Table 7-6 for $i = 0..Min(63, (1 \ll (4 + (sizeId \ll 1))) - 1)$.
- Otherwise, the scaling list is inferred from the reference scaling list as follows:

$$\text{refMatrixId} = \text{matrixId} - \text{scaling_list_pred_matrix_id_delta}[\text{sizeId}][\text{matrixId}] * (\text{sizeId} == 3 ? 3 : 1) \quad (7-42)$$

$$\text{ScalingList}[\text{sizeId}][\text{matrixId}][i] = \text{ScalingList}[\text{sizeId}][\text{refMatrixId}][i]$$

with $i = 0..Min(63, (1 \ll (4 + (sizeId \ll 1))) - 1)$ (7-43)

If `sizeId` is less than or equal to 2, the value of `scaling_list_pred_matrix_id_delta[sizeId][matrixId]` shall be in the range of 0 to `matrixId`, inclusive. Otherwise (`sizeId` is equal to 3), the value of `scaling_list_pred_matrix_id_delta[sizeId][matrixId]` shall be in the range of 0 to `matrixId / 3`, inclusive.

Table 7-3 – Specification of sizeId

Size of quantization matrix	sizeId
4x4	0
8x8	1
16x16	2
32x32	3

Table 7-4 – Specification of matrixId according to sizeId, prediction mode and colour component

sizeId	CuPredMode	cIdx (Colour component)	matrixId
0, 1, 2, 3	MODE_INTRA	0 (Y)	0
0, 1, 2, 3	MODE_INTRA	1 (Cb)	1
0, 1, 2, 3	MODE_INTRA	2 (Cr)	2
0, 1, 2, 3	MODE_INTER	0 (Y)	3
0, 1, 2, 3	MODE_INTER	1 (Cb)	4
0, 1, 2, 3	MODE_INTER	2 (Cr)	5

`scaling_list_dc_coef_minus8[sizeId - 2][matrixId]` plus 8 specifies the value of the variable `ScalingFactor[2][matrixId][0][0]` for the scaling list for the 16x16 size when `sizeId` is equal to 2 and specifies the value of `ScalingFactor[3][matrixId][0][0]` for the scaling list for the 32x32 size when `sizeId` is equal to 3. The value of `scaling_list_dc_coef_minus8[sizeId - 2][matrixId]` shall be in the range of -7 to 247, inclusive.

When `scaling_list_pred_mode_flag[sizeId][matrixId]` is equal to 0, `scaling_list_pred_matrix_id_delta[sizeId][matrixId]` is equal to 0 and `sizeId` is greater than 1, the value of `scaling_list_dc_coef_minus8[sizeId - 2][matrixId]` is inferred to be equal to 8.

When `scaling_list_pred_matrix_id_delta[sizeId][matrixId]` is not equal to 0 and `sizeId` is greater than 1, the value of `scaling_list_dc_coef_minus8[sizeId - 2][matrixId]` is inferred to be equal to `scaling_list_dc_coef_minus8[sizeId - 2][refMatrixId]`, where the value of `refMatrixId` is given by Equation 7-42.

`scaling_list_delta_coef` specifies the difference between the current matrix coefficient `ScalingList[sizeId][matrixId][i]` and the previous matrix coefficient `ScalingList[sizeId][matrixId][i - 1]`, when `scaling_list_pred_mode_flag[sizeId][matrixId]` is equal to 1. The value of `scaling_list_delta_coef` shall be in the range of -128 to 127, inclusive. The value of `ScalingList[sizeId][matrixId][i]` shall be greater than 0.

Table 7-5 – Specification of default values of ScalingList[0][matrixId][i] with i = 0..15

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ScalingList[0][0..5][i]	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16

Table 7-6 – Specification of default values of ScalingList[1..3][matrixId][i] with i = 0..63

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ScalingList[1..3][0..2][i]	16	16	16	16	16	16	16	16	16	16	17	16	17	16	17	18
ScalingList[1..3][3..5][i]	16	16	16	16	16	16	16	16	16	16	17	17	17	17	17	18
i – 16	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ScalingList[1..3][0..2][i]	17	18	18	17	18	21	19	20	21	20	19	21	24	22	22	24
ScalingList[1..3][3..5][i]	18	18	18	18	18	20	20	20	20	20	20	20	24	24	24	24
i – 32	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ScalingList[1..3][0..2][i]	24	22	22	24	25	25	27	30	27	25	25	29	31	35	35	31
ScalingList[1..3][3..5][i]	24	24	24	24	25	25	25	25	25	25	25	28	28	28	28	28
i – 48	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ScalingList[1..3][0..2][i]	29	36	41	44	41	36	47	54	54	47	65	70	65	88	88	115
ScalingList[1..3][3..5][i]	28	33	33	33	33	33	41	41	41	41	54	54	54	71	71	91

The four-dimensional array ScalingFactor[sizeId][matrixId][x][y], with $x, y = 0..(1 \ll (2 + \text{sizeId})) - 1$, specifies the array of scaling factors according to the variables sizeId specified in Table 7-3 and matrixId specified in Table 7-4.

The elements of the quantization matrix of size 4x4, ScalingFactor[0][matrixId][][], are derived as follows:

$$\text{ScalingFactor}[0][\text{matrixId}][x][y] = \text{ScalingList}[0][\text{matrixId}][i] \quad (7-44)$$

with $i = 0..15$, $\text{matrixId} = 0..5$, $x = \text{ScanOrder}[2][0][i][0]$, and $y = \text{ScanOrder}[2][0][i][1]$

The elements of the quantization matrix of size 8x8, ScalingFactor[1][matrixId][][], are derived as follows:

$$\text{ScalingFactor}[1][\text{matrixId}][x][y] = \text{ScalingList}[1][\text{matrixId}][i] \quad (7-45)$$

with $i = 0..63$, $\text{matrixId} = 0..5$, $x = \text{ScanOrder}[3][0][i][0]$, and $y = \text{ScanOrder}[3][0][i][1]$

The elements of the quantization matrix of size 16x16, ScalingFactor[2][matrixId][][], are derived as follows:

$$\text{ScalingFactor}[2][\text{matrixId}][x * 2 + k][y * 2 + j] = \text{ScalingList}[2][\text{matrixId}][i] \quad (7-46)$$

with $i = 0..63$, $j = 0..1$, $k = 0..1$, $\text{matrixId} = 0..5$, $x = \text{ScanOrder}[3][0][i][0]$, and $y = \text{ScanOrder}[3][0][i][1]$

$$\text{ScalingFactor}[2][\text{matrixId}][0][0] = \text{scaling_list_dc_coef_minus8}[0][\text{matrixId}] + 8 \quad (7-47)$$

with $\text{matrixId} = 0..5$

The elements of the quantization matrix of size 32x32, ScalingFactor[3][matrixId][][], are derived as follows:

$$\text{ScalingFactor}[3][\text{matrixId}][x * 4 + k][y * 4 + j] = \text{ScalingList}[3][\text{matrixId}][i] \quad (7-48)$$

with $i = 0..63$, $j = 0..3$, $k = 0..3$, $\text{matrixId} = 0, 3$, $x = \text{ScanOrder}[3][0][i][0]$, and $y = \text{ScanOrder}[3][0][i][1]$

$$\text{ScalingFactor}[3][\text{matrixId}][0][0] = \text{scaling_list_dc_coef_minus8}[1][\text{matrixId}] + 8 \quad (7-49)$$

with $\text{matrixId} = 0, 3$

When ChromaArrayType is equal to 3, the elements of the chroma quantization matrix of size 32x32, ScalingFactor[3][matrixId][][], with $\text{matrixId} = 1, 2, 4$ and 5 are derived as follows:

$$\text{ScalingFactor}[3][\text{matrixId}][x * 4 + k][y * 4 + j] = \text{ScalingList}[2][\text{matrixId}][i] \quad (7-50)$$

with $i = 0..63$, $j = 0..3$, $k = 0..3$, $x = \text{ScanOrder}[3][0][i][0]$, and $y = \text{ScanOrder}[3][0][i][1]$

$$\text{ScalingFactor}[3][\text{matrixId}][0][0] = \text{scaling_list_dc_coef_minus8}[0][\text{matrixId}] + 8 \quad (7-51)$$

7.4.6 Supplemental enhancement information message semantics

Each SEI message consists of the variables specifying the type `payloadType` and size `payloadSize` of the SEI message payload. SEI message payloads are specified in Annex D. The derived SEI message payload size `payloadSize` is specified in bytes and shall be equal to the number of RBSP bytes in the SEI message payload.

NOTE – The NAL unit byte sequence containing the SEI message might include one or more emulation prevention bytes (represented by `emulation_prevention_three_byte` syntax elements). Since the payload size of an SEI message is specified in RBSP bytes, the quantity of emulation prevention bytes is not included in the size `payloadSize` of an SEI payload.

ff_byte is a byte equal to 0xFF identifying a need for a longer representation of the syntax structure that it is used within.

last_payload_type_byte is the last byte of the payload type of an SEI message.

last_payload_size_byte is the last byte of the payload size of an SEI message.

7.4.7 Slice segment header semantics

7.4.7.1 General slice segment header semantics

When present, the value of the slice segment header syntax elements `slice_pic_parameter_set_id`, `pic_output_flag`, `no_output_of_prior_pics_flag`, `slice_pic_order_cnt_lsb`, `short_term_ref_pic_set_sps_flag`, `short_term_ref_pic_set_idx`, `num_long_term_sps`, `num_long_term_pics` and `slice_temporal_mvp_enabled_flag` shall be the same in all slice segment headers of a coded picture. When present, the value of the slice segment header syntax elements `lt_idx_sps[i]`, `poc_lsb_lt[i]`, `used_by_curr_pic_lt_flag[i]`, `delta_poc_msb_present_flag[i]` and `delta_poc_msb_cycle_lt[i]` shall be the same in all slice segment headers of a coded picture for each possible value of *i*.

first_slice_segment_in_pic_flag equal to 1 specifies that the slice segment is the first slice segment of the picture in decoding order. **first_slice_segment_in_pic_flag** equal to 0 specifies that the slice segment is not the first slice segment of the picture in decoding order.

NOTE 1 – This syntax element may be used for detection of the boundary between coded pictures that are consecutive in decoding order. However, when IDR pictures are consecutive in decoding order and have the same NAL unit type, loss of the first slice of an IDR picture can cause a problem with detection of the boundary between the coded pictures. This can occur, e.g., in the transmission of all-intra-coded video in an error-prone environment. This problem can be mitigated by alternately using the two different IDR NAL unit types (IDR_W_RADL and IDR_N_LP) for any two consecutive IDR pictures. The use of the temporal sub-layer zero index SEI message can also be helpful, as that SEI message includes the syntax element `irap_pic_id`, the value of which is different for IRAP pictures that are consecutive in decoding order. Some system environments have other provisions that can be helpful for picture boundary detection as well, such as the use of presentation timestamps in Rec. ITU-T H.222.0 | ISO/IEC 13818-1 systems, access unit framing in the ISO/IEC 14496-12 ISO base media file format, or the marker bit in IETF RFC 3550 real-time transport protocol headers.

no_output_of_prior_pics_flag affects the output of previously-decoded pictures in the decoded picture buffer after the decoding of an IDR or a BLA picture that is not the first picture in the bitstream as specified in Annex C.

slice_pic_parameter_set_id specifies the value of `pps_pic_parameter_set_id` for the PPS in use. The value of `slice_pic_parameter_set_id` shall be in the range of 0 to 63, inclusive.

Let `activePPS` be the PPS that has `pps_pic_parameter_set_id` equal to `slice_pic_parameter_set_id`, and let `activeSPS` be the SPS that has `sps_seq_parameter_set_id` equal to `pps_seq_parameter_set_id` of `activePPS`. It is a requirement of bitstream conformance that the following constraints apply:

- The value of `TemporalId` of `activePPS` shall be less than or equal to the value of `TemporalId` of the current picture.
- The value of `nuh_layer_id` of `activePPS` shall be less than or equal to the value of `nuh_layer_id` of the current picture.
- The value of `nuh_layer_id` of `activeSPS` shall be less than or equal to the value of `nuh_layer_id` of the current picture.

dependent_slice_segment_flag equal to 1 specifies that the value of each slice segment header syntax element that is not present is inferred to be equal to the value of the corresponding slice segment header syntax element in the slice header. When not present, the value of `dependent_slice_segment_flag` is inferred to be equal to 0.

The variable `SliceAddrRs` is derived as follows:

- If `dependent_slice_segment_flag` is equal to 0, `SliceAddrRs` is set equal to `slice_segment_address`.
- Otherwise, `SliceAddrRs` is set equal to `SliceAddrRs` of the preceding slice segment containing the CTB for which the CTB address is `CtbAddrTsToRs[CtbAddrRsToTs[slice_segment_address] – 1]`.

slice_segment_address specifies the address of the first CTB in the slice segment, in CTB raster scan of a picture. The length of the `slice_segment_address` syntax element is $\text{Ceil}(\text{Log}_2(\text{PicSizeInCtbsY}))$ bits. The value of

slice_segment_address shall be in the range of 0 to PicSizeInCtbsY – 1, inclusive, and the value of slice_segment_address shall not be equal to the value of slice_segment_address of any other coded slice segment NAL unit of the same coded picture. When slice_segment_address is not present, it is inferred to be equal to 0.

The variable CtbAddrInRs, specifying a CTB address in CTB raster scan of a picture, is set equal to slice_segment_address. The variable CtbAddrInTs, specifying a CTB address in tile scan, is set equal to CtbAddrRsToTs[CtbAddrInRs]. The variables CuQpOffsetCb and CuQpOffsetCr, specifying values to be used when determining the respective values of the Qp'Cb and Qp'Cr quantization parameters for the coding unit containing cu_chroma_qp_offset_flag, are both set equal to 0.

slice_reserved_flag[i] has semantics and values that are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the presence and value of slice_reserved_flag[i].

slice_type specifies the coding type of the slice according to Table 7-7.

Table 7-7 – Name association to slice_type

slice_type	Name of slice_type
0	B (B slice)
1	P (P slice)
2	I (I slice)

When nal_unit_type has a value in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive, i.e., the picture is an IRAP picture, nuh_layer_id is equal to 0, and pps_curr_pic_ref_enabled_flag is equal to 0, slice_type shall be equal to 2.

When sps_max_dec_pic_buffering_minus1[TemporalId] is equal to 0, nuh_layer_id is equal to 0, and pps_curr_pic_ref_enabled_flag is equal to 0, slice_type shall be equal to 2.

pic_output_flag affects the decoded picture output and removal processes as specified in Annex C. When pic_output_flag is not present, it is inferred to be equal to 1.

colour_plane_id specifies the colour plane associated with the current slice RBSP when separate_colour_plane_flag is equal to 1. The value of colour_plane_id shall be in the range of 0 to 2, inclusive. colour_plane_id values 0, 1 and 2 correspond to the Y, Cb and Cr planes, respectively.

NOTE 2 – There is no dependency between the decoding processes of pictures having different values of colour_plane_id.

slice_pic_order_cnt_lsb specifies the picture order count modulo MaxPicOrderCntLsb for the current picture. The length of the slice_pic_order_cnt_lsb syntax element is log2_max_pic_order_cnt_lsb_minus4 + 4 bits. The value of the slice_pic_order_cnt_lsb shall be in the range of 0 to MaxPicOrderCntLsb – 1, inclusive. When slice_pic_order_cnt_lsb is not present, slice_pic_order_cnt_lsb is inferred to be equal to 0, except as specified in clause 8.3.3.1.

short_term_ref_pic_set_sps_flag equal to 1 specifies that the short-term RPS of the current picture is derived based on one of the st_ref_pic_set() syntax structures in the active SPS that is identified by the syntax element short_term_ref_pic_set_idx in the slice header. short_term_ref_pic_set_sps_flag equal to 0 specifies that the short-term RPS of the current picture is derived based on the st_ref_pic_set() syntax structure that is directly included in the slice headers of the current picture. When num_short_term_ref_pic_sets is equal to 0, the value of short_term_ref_pic_set_sps_flag shall be equal to 0.

short_term_ref_pic_set_idx specifies the index, into the list of the st_ref_pic_set() syntax structures included in the active SPS, of the st_ref_pic_set() syntax structure that is used for derivation of the short-term RPS of the current picture. The syntax element short_term_ref_pic_set_idx is represented by Ceil(Log2(num_short_term_ref_pic_sets)) bits. When not present, the value of short_term_ref_pic_set_idx is inferred to be equal to 0. The value of short_term_ref_pic_set_idx shall be in the range of 0 to num_short_term_ref_pic_sets – 1, inclusive.

The variable CurrRpsIdx is derived as follows:

- If short_term_ref_pic_set_sps_flag is equal to 1, CurrRpsIdx is set equal to short_term_ref_pic_set_idx.
- Otherwise, CurrRpsIdx is set equal to num_short_term_ref_pic_sets.

num_long_term_sps specifies the number of entries in the long-term RPS of the current picture that are derived based on the candidate long-term reference pictures specified in the active SPS. The value of num_long_term_sps shall be in the range of 0 to num_long_term_ref_pics_sps, inclusive. When not present, the value of num_long_term_sps is inferred to be equal to 0.

num_long_term_pics specifies the number of entries in the long-term RPS of the current picture that are directly signalled in the slice header. When not present, the value of num_long_term_pics is inferred to be equal to 0.

When `nuh_layer_id` is equal to 0, the value of `num_long_term_pics` shall be less than or equal to `sps_max_dec_pic_buffering_minus1[TemporalId] – NumNegativePics[CurrRpsIdx] – NumPositivePics[CurrRpsIdx] – num_long_term_sps – TwoVersionsOfCurrDecPicFlag`.

`lt_idx_sps[i]` specifies an index, into the list of candidate long-term reference pictures specified in the active SPS, of the *i*-th entry in the long-term RPS of the current picture. The number of bits used to represent `lt_idx_sps[i]` is equal to $\text{Ceil}(\text{Log2}(\text{num_long_term_ref_pics_sps}))$. When not present, the value of `lt_idx_sps[i]` is inferred to be equal to 0. The value of `lt_idx_sps[i]` shall be in the range of 0 to `num_long_term_ref_pics_sps – 1`, inclusive.

`poc_lsb_lt[i]` specifies the value of the picture order count modulo `MaxPicOrderCntLsb` of the *i*-th entry in the long-term RPS of the current picture. The length of the `poc_lsb_lt[i]` syntax element is $\text{log2_max_pic_order_cnt_lsb_minus4} + 4$ bits.

`used_by_curr_pic_lt_flag[i]` equal to 0 specifies that the *i*-th entry in the long-term RPS of the current picture is not used for reference by the current picture.

The variables `PocLsbLt[i]` and `UsedByCurrPicLt[i]` are derived as follows:

- If *i* is less than `num_long_term_sps`, `PocLsbLt[i]` is set equal to `lt_ref_pic_poc_lsb_sps[lt_idx_sps[i]]` and `UsedByCurrPicLt[i]` is set equal to `used_by_curr_pic_lt_sps_flag[lt_idx_sps[i]]`.
- Otherwise, `PocLsbLt[i]` is set equal to `poc_lsb_lt[i]` and `UsedByCurrPicLt[i]` is set equal to `used_by_curr_pic_lt_flag[i]`.

`delta_poc_msb_present_flag[i]` equal to 1 specifies that `delta_poc_msb_cycle_lt[i]` is present. `delta_poc_msb_present_flag[i]` equal to 0 specifies that `delta_poc_msb_cycle_lt[i]` is not present.

Let `prevTid0Pic` be the previous picture in decoding order that has `TemporalId` equal to 0 and is not a RASL, RADL or SLNR picture. Let `setOfPrevPocVals` be a set consisting of the following:

- the `PicOrderCntVal` of `prevTid0Pic`,
- the `PicOrderCntVal` of each picture in the RPS of `prevTid0Pic`,
- the `PicOrderCntVal` of each picture that follows `prevTid0Pic` in decoding order and precedes the current picture in decoding order.

When there is more than one value in `setOfPrevPocVals` for which the value modulo `MaxPicOrderCntLsb` is equal to `PocLsbLt[i]`, `delta_poc_msb_present_flag[i]` shall be equal to 1.

`delta_poc_msb_cycle_lt[i]` is used to determine the value of the most significant bits of the picture order count value of the *i*-th entry in the long-term RPS of the current picture. The value of `delta_poc_msb_cycle_lt[i]` shall be in the range of 0 to $2^{(32 - \text{log2_max_pic_order_cnt_lsb_minus4} - 4)}$, inclusive. When `delta_poc_msb_cycle_lt[i]` is not present, it is inferred to be equal to 0.

The variable `DeltaPocMsbCycleLt[i]` is derived as follows:

$$\begin{aligned} &\text{if}(i == 0 \mid \mid i == \text{num_long_term_sps}) \\ &\quad \text{DeltaPocMsbCycleLt}[i] = \text{delta_poc_msb_cycle_lt}[i] \\ &\text{else} \\ &\quad \text{DeltaPocMsbCycleLt}[i] = \text{delta_poc_msb_cycle_lt}[i] + \text{DeltaPocMsbCycleLt}[i - 1] \end{aligned} \quad (7-52)$$

`slice_temporal_mvp_enabled_flag` specifies whether temporal motion vector predictors can be used for inter prediction. If `slice_temporal_mvp_enabled_flag` is equal to 0, the syntax elements of the current picture shall be constrained such that no temporal motion vector predictor is used in decoding of the current picture. Otherwise (`slice_temporal_mvp_enabled_flag` is equal to 1), temporal motion vector predictors may be used in decoding of the current picture. When not present, the value of `slice_temporal_mvp_enabled_flag` is inferred to be equal to 0.

Let `currLayerId` be equal to `nuh_layer_id` of the current NAL unit. When both `slice_temporal_mvp_enabled_flag` and `TemporalId` are equal to 0, the syntax elements for all coded pictures with `nuh_layer_id` equal to `currLayerId` that follow the current picture in decoding order shall be constrained such that no temporal motion vector from any picture with `nuh_layer_id` equal to `currLayerId` that precedes the current picture in decoding order is used in decoding of any coded picture that follows the current picture in decoding order.

NOTE 3 – When `slice_temporal_mvp_enabled_flag` is equal to 0 in an I slice, it has no impact on the normative decoding process of the picture but merely expresses a bitstream constraint.

NOTE 4 – When `slice_temporal_mvp_enabled_flag` is equal to 0 in a slice with `TemporalId` equal to 0, decoders may empty "motion vector storage" for all reference pictures with `nuh_layer_id` equal to `currLayerId` in the decoded picture buffer.

slice_sao_luma_flag equal to 1 specifies that SAO is enabled for the luma component in the current slice; slice_sao_luma_flag equal to 0 specifies that SAO is disabled for the luma component in the current slice. When slice_sao_luma_flag is not present, it is inferred to be equal to 0.

slice_sao_chroma_flag equal to 1 specifies that SAO is enabled for the chroma component in the current slice; slice_sao_chroma_flag equal to 0 specifies that SAO is disabled for the chroma component in the current slice. When slice_sao_chroma_flag is not present, it is inferred to be equal to 0.

num_ref_idx_active_override_flag equal to 1 specifies that the syntax element num_ref_idx_l0_active_minus1 is present for P and B slices and that the syntax element num_ref_idx_l1_active_minus1 is present for B slices. num_ref_idx_active_override_flag equal to 0 specifies that the syntax elements num_ref_idx_l0_active_minus1 and num_ref_idx_l1_active_minus1 are not present.

num_ref_idx_l0_active_minus1 specifies the maximum reference index for reference picture list 0 that may be used to decode the slice. num_ref_idx_l0_active_minus1 shall be in the range of 0 to 14, inclusive. When the current slice is a P or B slice and num_ref_idx_l0_active_minus1 is not present, num_ref_idx_l0_active_minus1 is inferred to be equal to num_ref_idx_l0_default_active_minus1.

num_ref_idx_l1_active_minus1 specifies the maximum reference index for reference picture list 1 that may be used to decode the slice. num_ref_idx_l1_active_minus1 shall be in the range of 0 to 14, inclusive. When num_ref_idx_l1_active_minus1 is not present, num_ref_idx_l1_active_minus1 is inferred to be equal to num_ref_idx_l1_default_active_minus1.

mvd_l1_zero_flag equal to 1 indicates that the mvd_coding(x0, y0, 1) syntax structure is not parsed and MvdL1[x0][y0][compIdx] is set equal to 0 for compIdx = 0..1. mvd_l1_zero_flag equal to 0 indicates that the mvd_coding(x0, y0, 1) syntax structure is parsed.

cabac_init_flag specifies the method for determining the initialization table used in the initialization process for context variables. When cabac_init_flag is not present, it is inferred to be equal to 0.

collocated_from_l0_flag equal to 1 specifies that the collocated picture used for temporal motion vector prediction is derived from reference picture list 0. collocated_from_l0_flag equal to 0 specifies that the collocated picture used for temporal motion vector prediction is derived from reference picture list 1. When collocated_from_l0_flag is not present, it is inferred to be equal to 1.

collocated_ref_idx specifies the reference index of the collocated picture used for temporal motion vector prediction.

When slice_type is equal to P or when slice_type is equal to B and collocated_from_l0_flag is equal to 1, collocated_ref_idx refers to a picture in list 0, and the value of collocated_ref_idx shall be in the range of 0 to num_ref_idx_l0_active_minus1, inclusive.

When slice_type is equal to B and collocated_from_l0_flag is equal to 0, collocated_ref_idx refers to a picture in list 1 and the value of collocated_ref_idx shall be in the range of 0 to num_ref_idx_l1_active_minus1, inclusive.

When not present, the value of collocated_ref_idx is inferred to be equal to 0.

When slice_temporal_mvp_enabled_flag is equal to 1, it is a requirement of bitstream conformance that, for all slices of the current picture that have slice_type not equal to 2 (if any), the picture referred to by collocated_ref_idx shall be the same and shall not be the current picture.

NOTE 5 – This implies that when pps_curr_pic_ref_enabled_flag is equal to 1 and the current picture is the only reference picture in the reference picture list, slice_temporal_mvp_enabled_flag would be constrained to be equal to 0.

five_minus_max_num_merge_cand specifies the maximum number of merging motion vector prediction (MVP) candidates supported in the slice subtracted from 5. The maximum number of merging MVP candidates, MaxNumMergeCand is derived as follows:

$$\text{MaxNumMergeCand} = 5 - \text{five_minus_max_num_merge_cand} \quad (7-53)$$

The value of MaxNumMergeCand shall be in the range of 1 to 5, inclusive.

use_integer_mv_flag equal to 1 specifies that the resolution of motion vectors for inter prediction in the current slice is integer. use_integer_mv_flag equal to 0 specifies that the resolution of motion vectors for inter prediction in the current slice that refer to pictures other than the current picture is fractional with quarter-sample precision in units of luma samples. When not present, the value of use_integer_mv_flag is inferred to be equal to motion_vector_resolution_control_idc.

slice_qp_delta specifies the initial value of Qp_Y to be used for the coding blocks in the slice until modified by the value of CuQpDeltaVal in the coding unit layer. The initial value of the Qp_Y quantization parameter for the slice, SliceQp_Y, is derived as follows:

$$\text{SliceQp}_Y = 26 + \text{init_qp_minus26} + \text{slice_qp_delta} \quad (7-54)$$

The value of SliceQp_Y shall be in the range of $-\text{QpBdOffset}_Y$ to +51, inclusive.

slice_cb_qp_offset specifies a difference to be added to the value of pps_cb_qp_offset when determining the value of the Qp'_{Cb} quantization parameter. The value of $\text{slice_cb_qp_offset}$ shall be in the range of -12 to $+12$, inclusive. When $\text{slice_cb_qp_offset}$ is not present, it is inferred to be equal to 0. The value of $\text{pps_cb_qp_offset} + \text{slice_cb_qp_offset}$ shall be in the range of -12 to $+12$, inclusive.

slice_cr_qp_offset specifies a difference to be added to the value of pps_cr_qp_offset when determining the value of the Qp'_{Cr} quantization parameter. The value of $\text{slice_cr_qp_offset}$ shall be in the range of -12 to $+12$, inclusive. When $\text{slice_cr_qp_offset}$ is not present, it is inferred to be equal to 0. The value of $\text{pps_cr_qp_offset} + \text{slice_cr_qp_offset}$ shall be in the range of -12 to $+12$, inclusive.

slice_act_y_qp_offset, **slice_act_cb_qp_offset** and **slice_act_cr_qp_offset** specify offsets to the quantization parameter values qP derived in clause 8.6.2 for luma, Cb, and Cr components, respectively. The values of $\text{slice_act_y_qp_offset}$, $\text{slice_act_cb_qp_offset}$ and $\text{slice_act_cr_qp_offset}$ shall be in the range of -12 to $+12$, inclusive. When not present, the values of $\text{slice_act_y_qp_offset}$, $\text{slice_act_cb_qp_offset}$, and $\text{slice_act_cr_qp_offset}$ are inferred to be equal to 0. The value of $\text{PpsActQpOffsetY} + \text{slice_act_y_qp_offset}$ shall be in the range of -12 to $+12$, inclusive. The value of $\text{PpsActQpOffsetCb} + \text{slice_act_cb_qp_offset}$ shall be in the range of -12 to $+12$, inclusive. The value of $\text{PpsActQpOffsetCr} + \text{slice_act_cr_qp_offset}$ shall be in the range of -12 to $+12$, inclusive.

cu_chroma_qp_offset_enabled_flag equal to 1 specifies that the $\text{cu_chroma_qp_offset_flag}$ may be present in the transform unit syntax. $\text{cu_chroma_qp_offset_enabled_flag}$ equal to 0 specifies that the $\text{cu_chroma_qp_offset_flag}$ is not present in the transform unit syntax. When not present, the value of $\text{cu_chroma_qp_offset_enabled_flag}$ is inferred to be equal to 0.

deblocking_filter_override_flag equal to 1 specifies that deblocking parameters are present in the slice header. $\text{deblocking_filter_override_flag}$ equal to 0 specifies that deblocking parameters are not present in the slice header. When not present, the value of $\text{deblocking_filter_override_flag}$ is inferred to be equal to 0.

slice_deblocking_filter_disabled_flag equal to 1 specifies that the operation of the deblocking filter is not applied for the current slice. $\text{slice_deblocking_filter_disabled_flag}$ equal to 0 specifies that the operation of the deblocking filter is applied for the current slice. When $\text{slice_deblocking_filter_disabled_flag}$ is not present, it is inferred to be equal to $\text{pps_deblocking_filter_disabled_flag}$.

slice_beta_offset_div2 and **slice_tc_offset_div2** specify the deblocking parameter offsets for β and tC (divided by 2) for the current slice. The values of $\text{slice_beta_offset_div2}$ and $\text{slice_tc_offset_div2}$ shall both be in the range of -6 to 6 , inclusive. When not present, the values of $\text{slice_beta_offset_div2}$ and $\text{slice_tc_offset_div2}$ are inferred to be equal to $\text{pps_beta_offset_div2}$ and $\text{pps_tc_offset_div2}$, respectively.

slice_loop_filter_across_slices_enabled_flag equal to 1 specifies that in-loop filtering operations may be performed across the left and upper boundaries of the current slice. $\text{slice_loop_filter_across_slices_enabled_flag}$ equal to 0 specifies that in-loop operations are not performed across left and upper boundaries of the current slice. The in-loop filtering operations include the deblocking filter and sample adaptive offset filter. When $\text{slice_loop_filter_across_slices_enabled_flag}$ is not present, it is inferred to be equal to $\text{pps_loop_filter_across_slices_enabled_flag}$.

num_entry_point_offsets specifies the number of $\text{entry_point_offset_minus1}[i]$ syntax elements in the slice header. When not present, the value of $\text{num_entry_point_offsets}$ is inferred to be equal to 0.

The value of $\text{num_entry_point_offsets}$ is constrained as follows:

- If $\text{tiles_enabled_flag}$ is equal to 0 and $\text{entropy_coding_sync_enabled_flag}$ is equal to 1, the value of $\text{num_entry_point_offsets}$ shall be in the range of 0 to $\text{PicHeightInCtbsY} - 1$, inclusive.
- Otherwise, if $\text{tiles_enabled_flag}$ is equal to 1 and $\text{entropy_coding_sync_enabled_flag}$ is equal to 0, the value of $\text{num_entry_point_offsets}$ shall be in the range of 0 to $(\text{num_tile_columns_minus1} + 1) * (\text{num_tile_rows_minus1} + 1) - 1$, inclusive.
- Otherwise, when $\text{tiles_enabled_flag}$ is equal to 1 and $\text{entropy_coding_sync_enabled_flag}$ is equal to 1, the value of $\text{num_entry_point_offsets}$ shall be in the range of 0 to $(\text{num_tile_columns_minus1} + 1) * \text{PicHeightInCtbsY} - 1$, inclusive.

offset_len_minus1 plus 1 specifies the length, in bits, of the $\text{entry_point_offset_minus1}[i]$ syntax elements. The value of offset_len_minus1 shall be in the range of 0 to 31, inclusive.

entry_point_offset_minus1[i] plus 1 specifies the i -th entry point offset in bytes, and is represented by $\text{offset_len_minus1} + 1$ bits. The slice segment data that follow the slice segment header consists of $\text{num_entry_point_offsets} + 1$ subsets, with subset index values ranging from 0 to $\text{num_entry_point_offsets}$, inclusive. The

first byte of the slice segment data is considered byte 0. When present, emulation prevention bytes that appear in the slice segment data portion of the coded slice segment NAL unit are counted as part of the slice segment data for purposes of subset identification. Subset 0 consists of bytes 0 to `entry_point_offset_minus1[0]`, inclusive, of the coded slice segment data, subset `k`, with `k` in the range of 1 to `num_entry_point_offsets - 1`, inclusive, consists of bytes `firstByte[k]` to `lastByte[k]`, inclusive, of the coded slice segment data with `firstByte[k]` and `lastByte[k]` defined as:

$$\text{firstByte}[k] = \sum_{n=1}^k (\text{entry_point_offset_minus1}[n-1] + 1) \quad (7-55)$$

$$\text{lastByte}[k] = \text{firstByte}[k] + \text{entry_point_offset_minus1}[k] \quad (7-56)$$

The last subset (with subset index equal to `num_entry_point_offsets`) consists of the remaining bytes of the coded slice segment data.

When `tiles_enabled_flag` is equal to 1 and `entropy_coding_sync_enabled_flag` is equal to 0, each subset shall consist of all coded bits of all CTUs in the slice segment that are within the same tile, and the number of subsets (i.e., the value of `num_entry_point_offsets + 1`) shall be equal to the number of tiles that contain CTUs that are in the coded slice segment.

NOTE 6 – When `tiles_enabled_flag` is equal to 1 and `entropy_coding_sync_enabled_flag` is equal to 0, each slice must include either a subset of the CTUs of one tile (in which case the syntax element `entry_point_offset_minus1[i]` is not present) or must include all CTUs of an integer number of complete tiles.

When `tiles_enabled_flag` is equal to 0 and `entropy_coding_sync_enabled_flag` is equal to 1, each subset `k` with `k` in the range of 0 to `num_entry_point_offsets`, inclusive, shall consist of all coded bits of all CTUs in the slice segment that include luma CTBs that are in the same luma CTB row of the picture, and the number of subsets (i.e., the value of `num_entry_point_offsets + 1`) shall be equal to the number of CTB rows of the picture that contain CTUs that are in the coded slice segment.

NOTE 7 – The last subset (i.e., subset `k` for `k` equal to `num_entry_point_offsets`) may or may not contain all CTUs that include luma CTBs that are in a luma CTB row of the picture.

When `tiles_enabled_flag` is equal to 1 and `entropy_coding_sync_enabled_flag` is equal to 1, each subset `k` with `k` in the range of 0 to `num_entry_point_offsets`, inclusive, shall consist of all coded bits of all CTUs in the slice segment that include luma CTBs that are in the same luma CTB row within a tile, and the number of subsets (i.e., the value of `num_entry_point_offsets + 1`) shall be equal to the number of luma CTB row scans in the tile scan for the CTUs of the coded slice segment.

slice_segment_header_extension_length specifies the length of the slice segment header extension data in bytes, not including the bits used for signalling **slice_segment_header_extension_length** itself. The value of **slice_segment_header_extension_length** shall be in the range of 0 to 256, inclusive. When not present, the value of **slice_segment_header_extension_length** is inferred to be equal to 0.

slice_segment_header_extension_data_byte may have any value. Decoders shall ignore the value of **slice_segment_header_extension_data_byte**. Its value does not affect the decoding process of the profiles specified in Annex A.

7.4.7.2 Reference picture list modification semantics

ref_pic_list_modification_flag_l0 equal to 1 indicates that reference picture list 0 is specified explicitly by a list of `list_entry_l0[i]` values. **ref_pic_list_modification_flag_l0** equal to 0 indicates that reference picture list 0 is determined implicitly. When **ref_pic_list_modification_flag_l0** is not present in the slice header, it is inferred to be equal to 0.

list_entry_l0[i] specifies the index of the reference picture in `RefPicListTemp0` to be placed at the current position of reference picture list 0. The length of the `list_entry_l0[i]` syntax element is $\text{Ceil}(\text{Log2}(\text{NumPicTotalCurr}))$ bits. The value of `list_entry_l0[i]` shall be in the range of 0 to `NumPicTotalCurr - 1`, inclusive. When the syntax element `list_entry_l0[i]` is not present in the slice header, it is inferred to be equal to 0.

The variable `NumPicTotalCurr` is derived as follows:

```

NumPicTotalCurr = 0
for( i = 0; i < NumNegativePics[ CurrRpsIdx ]; i++ )
    if( UsedByCurrPicS0[ CurrRpsIdx ][ i ] )
        NumPicTotalCurr++
for( i = 0; i < NumPositivePics[ CurrRpsIdx ]; i++ )
    if( UsedByCurrPicS1[ CurrRpsIdx ][ i ] )
        NumPicTotalCurr++
for( i = 0; i < num_long_term_sps + num_long_term_pics; i++ )

```

(7-57)

```

        if( UsedByCurrPicLt[ i ] )
            NumPicTotalCurr++
    if( pps_curr_pic_ref_enabled_flag )
        NumPicTotalCurr++

```

ref_pic_list_modification_flag_11 equal to 1 indicates that reference picture list 1 is specified explicitly by a list of **list_entry_11[i]** values. **ref_pic_list_modification_flag_11** equal to 0 indicates that reference picture list 1 is determined implicitly. When **ref_pic_list_modification_flag_11** is not present in the slice header, it is inferred to be equal to 0.

list_entry_11[i] specifies the index of the reference picture in **RefPicListTemp1** to be placed at the current position of reference picture list 1. The length of the **list_entry_11[i]** syntax element is $\text{Ceil}(\text{Log2}(\text{NumPicTotalCurr}))$ bits. The value of **list_entry_11[i]** shall be in the range of 0 to **NumPicTotalCurr** – 1, inclusive. When the syntax element **list_entry_11[i]** is not present in the slice header, it is inferred to be equal to 0.

7.4.7.3 Weighted prediction parameters semantics

luma_log2_weight_denom is the base 2 logarithm of the denominator for all luma weighting factors. The value of **luma_log2_weight_denom** shall be in the range of 0 to 7, inclusive.

delta_chroma_log2_weight_denom is the difference of the base 2 logarithm of the denominator for all chroma weighting factors. When **delta_chroma_log2_weight_denom** is not present, it is inferred to be equal to 0.

The variable **ChromaLog2WeightDenom** is derived to be equal to **luma_log2_weight_denom** + **delta_chroma_log2_weight_denom** and the value shall be in the range of 0 to 7, inclusive.

luma_weight_10_flag[i] equal to 1 specifies that weighting factors for the luma component of list 0 prediction using **RefPicList0[i]** are present. **luma_weight_10_flag[i]** equal to 0 specifies that these weighting factors are not present. When **luma_weight_10_flag[i]** is not present, it is inferred to be equal to 0.

chroma_weight_10_flag[i] equal to 1 specifies that weighting factors for the chroma prediction values of list 0 prediction using **RefPicList0[i]** are present. **chroma_weight_10_flag[i]** equal to 0 specifies that these weighting factors are not present. When **chroma_weight_10_flag[i]** is not present, it is inferred to be equal to 0.

delta_luma_weight_10[i] is the difference of the weighting factor applied to the luma prediction value for list 0 prediction using **RefPicList0[i]**.

The variable **LumaWeightL0[i]** is derived to be equal to $(1 \ll \text{luma_log2_weight_denom}) + \text{delta_luma_weight_10}[i]$. When **luma_weight_10_flag[i]** is equal to 1, the value of **delta_luma_weight_10[i]** shall be in the range of –128 to 127, inclusive. When **luma_weight_10_flag[i]** is equal to 0, **LumaWeightL0[i]** is inferred to be equal to $2^{\text{luma_log2_weight_denom}}$.

luma_offset_10[i] is the additive offset applied to the luma prediction value for list 0 prediction using **RefPicList0[i]**. The value of **luma_offset_10[i]** shall be in the range of $-\text{WpOffsetHalfRange}_Y$ to $\text{WpOffsetHalfRange}_Y - 1$, inclusive. When **luma_weight_10_flag[i]** is equal to 0, **luma_offset_10[i]** is inferred to be equal to 0.

delta_chroma_weight_10[i][j] is the difference of the weighting factor applied to the chroma prediction values for list 0 prediction using **RefPicList0[i]** with **j** equal to 0 for Cb and **j** equal to 1 for Cr.

The variable **ChromaWeightL0[i][j]** is derived to be equal to $(1 \ll \text{ChromaLog2WeightDenom}) + \text{delta_chroma_weight_10}[i][j]$. When **chroma_weight_10_flag[i]** is equal to 1, the value of **delta_chroma_weight_10[i][j]** shall be in the range of –128 to 127, inclusive. When **chroma_weight_10_flag[i]** is equal to 0, **ChromaWeightL0[i][j]** is inferred to be equal to $2^{\text{ChromaLog2WeightDenom}}$.

delta_chroma_offset_10[i][j] is the difference of the additive offset applied to the chroma prediction values for list 0 prediction using **RefPicList0[i]** with **j** equal to 0 for Cb and **j** equal to 1 for Cr.

The variable **ChromaOffsetL0[i][j]** is derived as follows:

$$\begin{aligned} \text{ChromaOffsetL0}[i][j] = & \text{Clip3}(-\text{WpOffsetHalfRange}_C, \text{WpOffsetHalfRange}_C - 1, \\ & (\text{WpOffsetHalfRange}_C + \text{delta_chroma_offset_10}[i][j] - \\ & ((\text{WpOffsetHalfRange}_C * \text{ChromaWeightL0}[i][j]) \gg \text{ChromaLog2WeightDenom}))) \end{aligned} \quad (7-58)$$

The value of **delta_chroma_offset_10[i][j]** shall be in the range of $-4 * \text{WpOffsetHalfRange}_C$ to $4 * \text{WpOffsetHalfRange}_C - 1$, inclusive. When **chroma_weight_10_flag[i]** is equal to 0, **ChromaOffsetL0[i][j]** is inferred to be equal to 0.

luma_weight_11_flag[i], **chroma_weight_11_flag[i]**, **delta_luma_weight_11[i]**, **luma_offset_11[i]**, **delta_chroma_weight_11[i][j]** and **delta_chroma_offset_11[i][j]** have the same semantics as **luma_weight_10_flag[i]**, **chroma_weight_10_flag[i]**, **delta_luma_weight_10[i]**, **luma_offset_10[i]**,

$\text{delta_chroma_weight_l0}[i][j]$ and $\text{delta_chroma_offset_l0}[i][j]$, respectively, with l0, L0, list 0 and List0 replaced by l1, L1, list 1 and List1, respectively.

The variable sumWeightL0Flags is derived to be equal to the sum of $\text{luma_weight_l0_flag}[i] + 2 * \text{chroma_weight_l0_flag}[i]$, for $i = 0.. \text{num_ref_idx_l0_active_minus1}$.

When slice_type is equal to B, the variable sumWeightL1Flags is derived to be equal to the sum of $\text{luma_weight_l1_flag}[i] + 2 * \text{chroma_weight_l1_flag}[i]$, for $i = 0.. \text{num_ref_idx_l1_active_minus1}$.

It is a requirement of bitstream conformance that, when slice_type is equal to P, sumWeightL0Flags shall be less than or equal to 24 and when slice_type is equal to B, the sum of sumWeightL0Flags and sumWeightL1Flags shall be less than or equal to 24.

7.4.8 Short-term reference picture set semantics

The $\text{st_ref_pic_set}(\text{stRpsIdx})$ syntax structure may be present in an SPS or in a slice header. Depending on whether the syntax structure is included in a slice header or an SPS, the following applies:

- If present in a slice header, the $\text{st_ref_pic_set}(\text{stRpsIdx})$ syntax structure specifies the short-term RPS of the current picture (the picture containing the slice), and the following applies:
 - The content of the $\text{st_ref_pic_set}(\text{stRpsIdx})$ syntax structure shall be the same in all slice headers of the current picture.
 - The value of stRpsIdx shall be equal to the syntax element $\text{num_short_term_ref_pic_sets}$ in the active SPS.
 - The short-term RPS of the current picture is also referred to as the $\text{num_short_term_ref_pic_sets}$ -th candidate short-term RPS in the semantics specified in the remainder of this clause.
- Otherwise (present in an SPS), the $\text{st_ref_pic_set}(\text{stRpsIdx})$ syntax structure specifies a candidate short-term RPS, and the term "the current picture" in the semantics specified in the remainder of this clause refers to each picture that has $\text{short_term_ref_pic_set_idx}$ equal to stRpsIdx in a CVS that has the SPS as the active SPS.

$\text{inter_ref_pic_set_prediction_flag}$ equal to 1 specifies that the stRpsIdx -th candidate short-term RPS is predicted from another candidate short-term RPS, which is referred to as the source candidate short-term RPS. When $\text{inter_ref_pic_set_prediction_flag}$ is not present, it is inferred to be equal to 0.

delta_idx_minus1 plus 1 specifies the difference between the value of stRpsIdx and the index, into the list of the candidate short-term RPSs specified in the SPS, of the source candidate short-term RPS. The value of delta_idx_minus1 shall be in the range of 0 to $\text{stRpsIdx} - 1$, inclusive. When delta_idx_minus1 is not present, it is inferred to be equal to 0.

The variable RefRpsIdx is derived as follows:

$$\text{RefRpsIdx} = \text{stRpsIdx} - (\text{delta_idx_minus1} + 1) \quad (7-59)$$

delta_rps_sign and $\text{abs_delta_rps_minus1}$ together specify the value of the variable deltaRps as follows:

$$\text{deltaRps} = (1 - 2 * \text{delta_rps_sign}) * (\text{abs_delta_rps_minus1} + 1) \quad (7-60)$$

The variable deltaRps represents the value to be added to the picture order count difference values of the source candidate short-term RPS to obtain the picture order count difference values of the stRpsIdx -th candidate short-term RPS. The value of $\text{abs_delta_rps_minus1}$ shall be in the range of 0 to $2^{15} - 1$, inclusive.

$\text{used_by_curr_pic_flag}[j]$ equal to 0 specifies that the j -th entry in the source candidate short-term RPS is not used for reference by the current picture.

$\text{use_delta_flag}[j]$ equal to 1 specifies that the j -th entry in the source candidate short-term RPS is included in the stRpsIdx -th candidate short-term RPS. $\text{use_delta_flag}[j]$ equal to 0 specifies that the j -th entry in the source candidate short-term RPS is not included in the stRpsIdx -th candidate short-term RPS. When $\text{use_delta_flag}[j]$ is not present, its value is inferred to be equal to 1.

When $\text{inter_ref_pic_set_prediction_flag}$ is equal to 1, the variables $\text{DeltaPocS0}[\text{stRpsIdx}][i]$, $\text{UsedByCurrPicS0}[\text{stRpsIdx}][i]$, $\text{NumNegativePics}[\text{stRpsIdx}]$, $\text{DeltaPocS1}[\text{stRpsIdx}][i]$, $\text{UsedByCurrPicS1}[\text{stRpsIdx}][i]$ and $\text{NumPositivePics}[\text{stRpsIdx}]$ are derived as follows:

```
i = 0
for( j = NumPositivePics[ RefRpsIdx ] - 1; j >= 0; j-- ) {
    dPoc = DeltaPocS1[ RefRpsIdx ][ j ] + deltaRps
```

```

        if( dPoc < 0 && use_delta_flag[ NumNegativePics[ RefRpsIdx ] + j ] ) {
            DeltaPocS0[ stRpsIdx ][ i ] = dPoc
            UsedByCurrPicS0[ stRpsIdx ][ i++ ] =
used_by_curr_pic_flag[ NumNegativePics[ RefRpsIdx ] + j ]
        }
    }
    if( deltaRps < 0 && use_delta_flag[ NumDeltaPocs[ RefRpsIdx ] ] ) {
        DeltaPocS0[ stRpsIdx ][ i ] = deltaRps
        UsedByCurrPicS0[ stRpsIdx ][ i++ ] = used_by_curr_pic_flag[ NumDeltaPocs[ RefRpsIdx ] ]
    }
    for( j = 0; j < NumNegativePics[ RefRpsIdx ]; j++ ) {
        dPoc = DeltaPocS0[ RefRpsIdx ][ j ] + deltaRps
        if( dPoc < 0 && use_delta_flag[ j ] ) {
            DeltaPocS0[ stRpsIdx ][ i ] = dPoc
            UsedByCurrPicS0[ stRpsIdx ][ i++ ] = used_by_curr_pic_flag[ j ]
        }
    }
    NumNegativePics[ stRpsIdx ] = i

    i = 0
    for( j = NumNegativePics[ RefRpsIdx ] - 1; j >= 0; j-- ) {
        dPoc = DeltaPocS0[ RefRpsIdx ][ j ] + deltaRps
        if( dPoc > 0 && use_delta_flag[ j ] ) {
            DeltaPocS1[ stRpsIdx ][ i ] = dPoc
            UsedByCurrPicS1[ stRpsIdx ][ i++ ] = used_by_curr_pic_flag[ j ]
        }
    }
    if( deltaRps > 0 && use_delta_flag[ NumDeltaPocs[ RefRpsIdx ] ] ) {
        DeltaPocS1[ stRpsIdx ][ i ] = deltaRps
        UsedByCurrPicS1[ stRpsIdx ][ i++ ] = used_by_curr_pic_flag[ NumDeltaPocs[ RefRpsIdx ] ]
    }
    for( j = 0; j < NumPositivePics[ RefRpsIdx ]; j++ ) {
        dPoc = DeltaPocS1[ RefRpsIdx ][ j ] + deltaRps
        if( dPoc > 0 && use_delta_flag[ NumNegativePics[ RefRpsIdx ] + j ] ) {
            DeltaPocS1[ stRpsIdx ][ i ] = dPoc
            UsedByCurrPicS1[ stRpsIdx ][ i++ ] =
used_by_curr_pic_flag[ NumNegativePics[ RefRpsIdx ] + j ]
        }
    }
    NumPositivePics[ stRpsIdx ] = i

```

num_negative_pics specifies the number of entries in the stRpsIdx-th candidate short-term RPS that have picture order count values less than the picture order count value of the current picture. When nuh_layer_id of the current picture is equal to 0, the value of num_negative_pics shall be in the range of 0 to sps_max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1], inclusive.

num_positive_pics specifies the number of entries in the stRpsIdx-th candidate short-term RPS that have picture order count values greater than the picture order count value of the current picture. When nuh_layer_id of the current picture is equal to 0, the value of num_positive_pics shall be in the range of 0 to sps_max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1] - num_negative_pics, inclusive.

delta_poc_s0_minus1[i] plus 1, when i is equal to 0, specifies the difference between the picture order count values of the current picture and the i-th entry in the stRpsIdx-th candidate short-term RPS that has picture order count value less than that of the current picture, or, when i is greater than 0, specifies the difference between the picture order count values of the (i - 1)-th entry and the i-th entry in the stRpsIdx-th candidate short-term RPS that have picture order count values less than the picture order count value of the current picture. The value of delta_poc_s0_minus1[i] shall be in the range of 0 to $2^{15} - 1$, inclusive.

used_by_curr_pic_s0_flag[i] equal to 0 specifies that the i-th entry in the stRpsIdx-th candidate short-term RPS that has picture order count value less than that of the current picture is not used for reference by the current picture.

delta_poc_s1_minus1[i] plus 1, when i is equal to 0, specifies the difference between the picture order count values of the current picture and the i-th entry in the stRpsIdx-th candidate short-term RPS that has picture order count value greater than that of the current picture, or, when i is greater than 0, specifies the difference between the picture order count values of the i-th entry and the (i – 1)-th entry in the current candidate short-term RPS that have picture order count values greater than the picture order count value of the current picture. The value of delta_poc_s1_minus1[i] shall be in the range of 0 to $2^{15} - 1$, inclusive.

used_by_curr_pic_s1_flag[i] equal to 0 specifies that the i-th entry in the current candidate short-term RPS that has picture order count value greater than that of the current picture is not used for reference by the current picture.

When **inter_ref_pic_set_prediction_flag** is equal to 0, the variables **NumNegativePics**[stRpsIdx], **NumPositivePics**[stRpsIdx], **UsedByCurrPicS0**[stRpsIdx][i], **UsedByCurrPicS1**[stRpsIdx][i], **DeltaPocS0**[stRpsIdx][i] and **DeltaPocS1**[stRpsIdx][i] are derived as follows:

$$\text{NumNegativePics}[\text{stRpsIdx}] = \text{num_negative_pics} \quad (7-63)$$

$$\text{NumPositivePics}[\text{stRpsIdx}] = \text{num_positive_pics} \quad (7-64)$$

$$\text{UsedByCurrPicS0}[\text{stRpsIdx}][i] = \text{used_by_curr_pic_s0_flag}[i] \quad (7-65)$$

$$\text{UsedByCurrPicS1}[\text{stRpsIdx}][i] = \text{used_by_curr_pic_s1_flag}[i] \quad (7-66)$$

– If i is equal to 0, the following applies:

$$\text{DeltaPocS0}[\text{stRpsIdx}][i] = -(\text{delta_poc_s0_minus1}[i] + 1) \quad (7-67)$$

$$\text{DeltaPocS1}[\text{stRpsIdx}][i] = \text{delta_poc_s1_minus1}[i] + 1 \quad (7-68)$$

– Otherwise, the following applies:

$$\text{DeltaPocS0}[\text{stRpsIdx}][i] = \text{DeltaPocS0}[\text{stRpsIdx}][i - 1] - (\text{delta_poc_s0_minus1}[i] + 1) \quad (7-69)$$

$$\text{DeltaPocS1}[\text{stRpsIdx}][i] = \text{DeltaPocS1}[\text{stRpsIdx}][i - 1] + (\text{delta_poc_s1_minus1}[i] + 1) \quad (7-70)$$

The variable **NumDeltaPocs**[stRpsIdx] is derived as follows:

$$\text{NumDeltaPocs}[\text{stRpsIdx}] = \text{NumNegativePics}[\text{stRpsIdx}] + \text{NumPositivePics}[\text{stRpsIdx}] \quad (7-71)$$

7.4.9 Slice segment data semantics

7.4.9.1 General slice segment data semantics

end_of_slice_segment_flag equal to 0 specifies that another CTU is following in the slice. **end_of_slice_segment_flag** equal to 1 specifies the end of the slice segment, i.e., that no further CTU follows in the slice segment.

end_of_subset_one_bit shall be equal to 1.

7.4.9.2 Coding tree unit semantics

The CTU is the root node of the coding quadtree structure.

7.4.9.3 Sample adaptive offset semantics

sao_merge_left_flag equal to 1 specifies that the syntax elements **sao_type_idx_luma**, **sao_type_idx_chroma**, **sao_band_position**, **sao_eo_class_luma**, **sao_eo_class_chroma**, **sao_offset_abs** and **sao_offset_sign** are derived from the corresponding syntax elements of the left CTB. **sao_merge_left_flag** equal to 0 specifies that these syntax elements are not derived from the corresponding syntax elements of the left CTB. When **sao_merge_left_flag** is not present, it is inferred to be equal to 0.

sao_merge_up_flag equal to 1 specifies that the syntax elements **sao_type_idx_luma**, **sao_type_idx_chroma**, **sao_band_position**, **sao_eo_class_luma**, **sao_eo_class_chroma**, **sao_offset_abs** and **sao_offset_sign** are derived from the corresponding syntax elements of the above CTB. **sao_merge_up_flag** equal to 0 specifies that these syntax elements are

not derived from the corresponding syntax elements of the above CTB. When `sao_merge_up_flag` is not present, it is inferred to be equal to 0.

sao_type_idx_luma specifies the offset type for the luma component. The array `SaoTypeIdx[cIdx][rx][ry]` specifies the offset type as specified in Table 7-8 for the CTB at the location (rx, ry) for the colour component cIdx. The value of `SaoTypeIdx[0][rx][ry]` is derived as follows:

- If `sao_type_idx_luma` is present, `SaoTypeIdx[0][rx][ry]` is set equal to `sao_type_idx_luma`.
- Otherwise (`sao_type_idx_luma` is not present), `SaoTypeIdx[0][rx][ry]` is derived as follows:
 - If `sao_merge_left_flag` is equal to 1, `SaoTypeIdx[0][rx][ry]` is set equal to `SaoTypeIdx[0][rx - 1][ry]`.
 - Otherwise, if `sao_merge_up_flag` is equal to 1, `SaoTypeIdx[0][rx][ry]` is set equal to `SaoTypeIdx[0][rx][ry - 1]`.
 - Otherwise, `SaoTypeIdx[0][rx][ry]` is set equal to 0.

sao_type_idx_chroma specifies the offset type for the chroma components. The values of `SaoTypeIdx[cIdx][rx][ry]` are derived as follows for cIdx equal to 1..2:

- If `sao_type_idx_chroma` is present, `SaoTypeIdx[cIdx][rx][ry]` is set equal to `sao_type_idx_chroma`.
- Otherwise (`sao_type_idx_chroma` is not present), `SaoTypeIdx[cIdx][rx][ry]` is derived as follows:
 - If `sao_merge_left_flag` is equal to 1, `SaoTypeIdx[cIdx][rx][ry]` is set equal to `SaoTypeIdx[cIdx][rx - 1][ry]`.
 - Otherwise, if `sao_merge_up_flag` is equal to 1, `SaoTypeIdx[cIdx][rx][ry]` is set equal to `SaoTypeIdx[cIdx][rx][ry - 1]`.
 - Otherwise, `SaoTypeIdx[cIdx][rx][ry]` is set equal to 0.

Table 7-8 – Specification of the SAO type

SaoTypeIdx[cIdx][rx][ry]	SAO type (informative)
0	Not applied
1	Band offset
2	Edge offset

sao_offset_abs[cIdx][rx][ry][i] specifies the offset value of i-th category for the CTB at the location (rx, ry) for the colour component cIdx.

When `sao_offset_abs[cIdx][rx][ry][i]` is not present, it is inferred as follows:

- If `sao_merge_left_flag` is equal to 1, `sao_offset_abs[cIdx][rx][ry][i]` is inferred to be equal to `sao_offset_abs[cIdx][rx - 1][ry][i]`.
- Otherwise, if `sao_merge_up_flag` is equal to 1, `sao_offset_abs[cIdx][rx][ry][i]` is inferred to be equal to `sao_offset_abs[cIdx][rx][ry - 1][i]`.
- Otherwise, `sao_offset_abs[cIdx][rx][ry][i]` is inferred to be equal to 0.

sao_offset_sign[cIdx][rx][ry][i] specifies the sign of the offset value of i-th category for the CTB at the location (rx, ry) for the colour component cIdx.

When `sao_offset_sign[cIdx][rx][ry][i]` is not present, it is inferred as follows:

- If `sao_merge_left_flag` is equal to 1, `sao_offset_sign[cIdx][rx][ry][i]` is inferred to be equal to `sao_offset_sign[cIdx][rx - 1][ry][i]`.
- Otherwise, if `sao_merge_up_flag` is equal to 1, `sao_offset_sign[cIdx][rx][ry][i]` is inferred to be equal to `sao_offset_sign[cIdx][rx][ry - 1][i]`.
- Otherwise, if `SaoTypeIdx[cIdx][rx][ry]` is equal to 2, the following applies:
 - If i is equal to 0 or 1, `sao_offset_sign[cIdx][rx][ry][i]` is inferred to be equal 0.
 - Otherwise (i is equal to 2 or 3), `sao_offset_sign[cIdx][rx][ry][i]` is inferred to be equal 1.
- Otherwise, `sao_offset_sign[cIdx][rx][ry][i]` is inferred to be equal 0.

The variable log2OffsetScale is derived as follows:

- If cIdx is equal to 0, log2OffsetScale is set equal to log2_sao_offset_scale_luma.
- Otherwise (cIdx is equal to 1 or 2), log2OffsetScale is set equal to log2_sao_offset_scale_chroma.

The list SaoOffsetVal[cIdx][rx][ry][i] for i ranging from 0 to 4, inclusive, is derived as follows:

$$\begin{aligned} \text{SaoOffsetVal}[\text{cIdx}][\text{rx}][\text{ry}][0] &= 0 \\ \text{for}(i = 0; i < 4; i++) \\ \text{SaoOffsetVal}[\text{cIdx}][\text{rx}][\text{ry}][i + 1] &= (1 - 2 * \text{sao_offset_sign}[\text{cIdx}][\text{rx}][\text{ry}][i]) * \\ &\quad \text{sao_offset_abs}[\text{cIdx}][\text{rx}][\text{ry}][i] << \text{log2OffsetScale} \end{aligned} \quad (7-72)$$

sao_band_position[cIdx][rx][ry] specifies the displacement of the band offset of the sample range when SaoTypeIdx[cIdx][rx][ry] is equal to 1.

When sao_band_position[cIdx][rx][ry] is not present, it is inferred as follows:

- If sao_merge_left_flag is equal to 1, sao_band_position[cIdx][rx][ry] is inferred to be equal to sao_band_position[cIdx][rx - 1][ry].
- Otherwise, if sao_merge_up_flag is equal to 1, sao_band_position[cIdx][rx][ry] is inferred to be equal to sao_band_position[cIdx][rx][ry - 1].
- Otherwise, sao_band_position[cIdx][rx][ry] is inferred to be equal to 0.

sao_eo_class_luma specifies the edge offset class for the luma component. The array SaoEoClass[cIdx][rx][ry] specifies the offset type as specified in Table 7-9 for the CTB at the location (rx, ry) for the colour component cIdx. The value of SaoEoClass[0][rx][ry] is derived as follows:

- If sao_eo_class_luma is present, SaoEoClass[0][rx][ry] is set equal to sao_eo_class_luma.
- Otherwise (sao_eo_class_luma is not present), SaoEoClass[0][rx][ry] is derived as follows:
 - If sao_merge_left_flag is equal to 1, SaoEoClass[0][rx][ry] is set equal to SaoEoClass[0][rx - 1][ry].
 - Otherwise, if sao_merge_up_flag is equal to 1, SaoEoClass[0][rx][ry] is set equal to SaoEoClass[0][rx][ry - 1].
 - Otherwise, SaoEoClass[0][rx][ry] is set equal to 0.

sao_eo_class_chroma specifies the edge offset class for the chroma components. The values of SaoEoClass[cIdx][rx][ry] are derived as follows for cIdx equal to 1..2:

- If sao_eo_class_chroma is present, SaoEoClass[cIdx][rx][ry] is set equal to sao_eo_class_chroma.
- Otherwise (sao_eo_class_chroma is not present), SaoEoClass[cIdx][rx][ry] is derived as follows:
 - If sao_merge_left_flag is equal to 1, SaoEoClass[cIdx][rx][ry] is set equal to SaoEoClass[cIdx][rx - 1][ry].
 - Otherwise, if sao_merge_up_flag is equal to 1, SaoEoClass[cIdx][rx][ry] is set equal to SaoEoClass[cIdx][rx][ry - 1].
 - Otherwise, SaoEoClass[cIdx][rx][ry] is set equal to 0.

Table 7-9 – Specification of the SAO edge offset class

SaoEoClass[cIdx][rx][ry]	SAO edge offset class (informative)
0	1D 0-degree edge offset
1	1D 90-degree edge offset
2	1D 135-degree edge offset
3	1D 45-degree edge offset

7.4.9.4 Coding quadtree semantics

split_cu_flag[x0][y0] specifies whether a coding unit is split into coding units with half horizontal and vertical size. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

When **split_cu_flag**[x0][y0] is not present, the following applies:

- If $\log_2\text{CbSize}$ is greater than MinCbLog2SizeY , the value of **split_cu_flag**[x0][y0] is inferred to be equal to 1.
- Otherwise ($\log_2\text{CbSize}$ is equal to MinCbLog2SizeY), the value of **split_cu_flag**[x0][y0] is inferred to be equal to 0.

The array **CtDepth**[x][y] specifies the coding tree depth for a luma coding block covering the location (x, y). When **split_cu_flag**[x0][y0] is equal to 0, **CtDepth**[x][y] is inferred to be equal to **cqtDepth** for $x = x0..x0 + \text{nCbS} - 1$ and $y = y0..y0 + \text{nCbS} - 1$.

7.4.9.5 Coding unit semantics

cu_transquant_bypass_flag equal to 1 specifies that the scaling and transform process as specified in clause 8.6 and the in-loop filter process as specified in clause 8.7 are bypassed. When **cu_transquant_bypass_flag** is not present, it is inferred to be equal to 0.

cu_skip_flag[x0][y0] equal to 1 specifies that for the current coding unit, when decoding a P or B slice, no more syntax elements except the merging candidate index **merge_idx**[x0][y0] are parsed after **cu_skip_flag**[x0][y0]. **cu_skip_flag**[x0][y0] equal to 0 specifies that the coding unit is not skipped. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

When **cu_skip_flag**[x0][y0] is not present, it is inferred to be equal to 0.

pred_mode_flag equal to 0 specifies that the current coding unit is coded in inter prediction mode. **pred_mode_flag** equal to 1 specifies that the current coding unit is coded in intra prediction mode. The variable **CuPredMode**[x][y] is derived as follows for $x = x0..x0 + \text{nCbS} - 1$ and $y = y0..y0 + \text{nCbS} - 1$:

- If **pred_mode_flag** is equal to 0, **CuPredMode**[x][y] is set equal to **MODE_INTER**.
- Otherwise (**pred_mode_flag** is equal to 1), **CuPredMode**[x][y] is set equal to **MODE_INTRA**.

When **pred_mode_flag** is not present, the variable **CuPredMode**[x][y] is derived as follows for $x = x0..x0 + \text{nCbS} - 1$ and $y = y0..y0 + \text{nCbS} - 1$:

- If **slice_type** is equal to I, **CuPredMode**[x][y] is inferred to be equal to **MODE_INTRA**.
- Otherwise (**slice_type** is equal to P or B), when **cu_skip_flag**[x0][y0] is equal to 1, **CuPredMode**[x][y] is inferred to be equal to **MODE_SKIP**.

palette_mode_flag[x0][y0] equal to 1 specifies that the current coding unit is coded using the palette mode. **palette_mode_flag**[x0][y0] equal to 0 specifies that the current coding unit is not coded using the palette mode. The array indices x0 and y0 specify the location (x0, y0) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

When **palette_mode_flag**[x0][y0] is not present, it is inferred to be equal to 0.

part_mode specifies the partitioning mode of the current coding unit. The semantics of **part_mode** depend on **CuPredMode**[x0][y0]. The variables **PartMode** and **IntraSplitFlag** are derived from the value of **part_mode** as defined in Table 7-10.

The value of **part_mode** is restricted as follows:

- If **CuPredMode**[x0][y0] is equal to **MODE_INTRA**, **part_mode** shall be equal to 0 or 1.
- Otherwise (**CuPredMode**[x0][y0] is equal to **MODE_INTER**), the following applies:
 - If $\log_2\text{CbSize}$ is greater than MinCbLog2SizeY and **amp_enabled_flag** is equal to 1, **part_mode** shall be in the range of 0 to 2, inclusive, or in the range of 4 to 7, inclusive.
 - Otherwise, if $\log_2\text{CbSize}$ is greater than MinCbLog2SizeY and **amp_enabled_flag** is equal to 0, or $\log_2\text{CbSize}$ is equal to 3, **part_mode** shall be in the range of 0 to 2, inclusive.
 - Otherwise ($\log_2\text{CbSize}$ is greater than 3 and equal to MinCbLog2SizeY), the value of **part_mode** shall be in the range of 0 to 3, inclusive.

When **part_mode** is not present, the variables **PartMode** and **IntraSplitFlag** are derived as follows:

- PartMode is set equal to PART_2Nx2N.
- IntraSplitFlag is set equal to 0.

pcm_flag[x0][y0] equal to 1 specifies that the pcm_sample() syntax structure is present and the transform_tree() syntax structure is not present in the coding unit including the luma coding block at the location (x0, y0). pcm_flag[x0][y0] equal to 0 specifies that pcm_sample() syntax structure is not present. When pcm_flag[x0][y0] is not present, it is inferred to be equal to 0.

The value of pcm_flag[x0 + i][y0 + j] with $i = 1..nC_{bS} - 1$, $j = 1..nC_{bS} - 1$ is inferred to be equal to pcm_flag[x0][y0].

pcm_alignment_zero_bit is a bit equal to 0.

Table 7-10 – Name association to prediction mode and partitioning type

CuPredMode[x0][y0]	part_mode	IntraSplitFlag	PartMode
MODE_INTRA	0	0	PART_2Nx2N
	1	1	PART_NxN
MODE_INTER	0	0	PART_2Nx2N
	1	0	PART_2NxN
	2	0	PART_Nx2N
	3	0	PART_NxN
	4	0	PART_2NxN_U
	5	0	PART_2NxN_D
	6	0	PART_nLx2N
	7	0	PART_nRx2N

The syntax elements **prev_intra_luma_pred_flag**[x0 + i][y0 + j], **mpm_idx**[x0 + i][y0 + j] and **rem_intra_luma_pred_mode**[x0 + i][y0 + j] specify the intra prediction mode for luma samples. The array indices x0 + i, y0 + j specify the location (x0 + i, y0 + j) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture. When prev_intra_luma_pred_flag[x0 + i][y0 + j] is equal to 1, the intra prediction mode is inferred from a neighbouring intra-predicted prediction unit according to clause 8.4.2.

intra_chroma_pred_mode[x0][y0] specifies the intra prediction mode for chroma samples. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

rqt_root_cbf equal to 1 specifies that the transform_tree() syntax structure is present for the current coding unit. rqt_root_cbf equal to 0 specifies that the transform_tree() syntax structure is not present for the current coding unit.

When rqt_root_cbf is not present, its value is inferred to be equal to 1.

7.4.9.6 Prediction unit semantics

mvp_l0_flag[x0][y0] specifies the motion vector predictor index of list 0 where x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

When mvp_l0_flag[x0][y0] is not present, it is inferred to be equal to 0.

mvp_l1_flag[x0][y0] has the same semantics as mvp_l0_flag, with l0 and list 0 replaced by l1 and list 1, respectively.

merge_flag[x0][y0] specifies whether the inter prediction parameters for the current prediction unit are inferred from a neighbouring inter-predicted partition. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

When merge_flag[x0][y0] is not present, it is inferred as follows:

- If CuPredMode[x0][y0] is equal to MODE_SKIP, merge_flag[x0][y0] is inferred to be equal to 1.
- Otherwise, merge_flag[x0][y0] is inferred to be equal to 0.

merge_idx[x0][y0] specifies the merging candidate index of the merging candidate list where x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

When merge_idx[x0][y0] is not present, it is inferred to be equal to 0.

inter_pred_idc[x0][y0] specifies whether list0, list1, or bi-prediction is used for the current prediction unit according to Table 7-11. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

Table 7-11 – Name association to inter prediction mode

inter_pred_idc	Name of inter_pred_idc	
	(nPbW + nPbH) != 12	(nPbW + nPbH) == 12
0	PRED_L0	PRED_L0
1	PRED_L1	PRED_L1
2	PRED_BI	na

When inter_pred_idc[x0][y0] is not present, it is inferred to be equal to PRED_L0.

ref_idx_l0[x0][y0] specifies the list 0 reference picture index for the current prediction unit. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

When ref_idx_l0[x0][y0] is not present it is inferred to be equal to 0.

ref_idx_l1[x0][y0] has the same semantics as ref_idx_l0, with l0 and list 0 replaced by l1 and list 1, respectively.

7.4.9.7 PCM sample semantics

pcm_sample_luma[i] represents a coded luma sample value in the raster scan within the coding unit. The number of bits used to represent each of these samples is PcmBitDepth_Y.

pcm_sample_chroma[i] represents a coded chroma sample value in the raster scan within the coding unit. The first half of the values represent coded Cb samples and the remaining half of the values represent coded Cr samples. The number of bits used to represent each of these samples is PcmBitDepth_C.

7.4.9.8 Transform tree semantics

split_transform_flag[x0][y0][trafoDepth] specifies whether a block is split into four blocks with half horizontal and half vertical size for the purpose of transform coding. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered block relative to the top-left luma sample of the picture. The array index trafoDepth specifies the current subdivision level of a coding block into blocks for the purpose of transform coding. trafoDepth is equal to 0 for blocks that correspond to coding blocks.

The variable interSplitFlag is derived as follows:

- If max_transform_hierarchy_depth_inter is equal to 0 and CuPredMode[x0][y0] is equal to MODE_INTER and PartMode is not equal to PART_2Nx2N and trafoDepth is equal to 0, interSplitFlag is set equal to 1.
- Otherwise, interSplitFlag is set equal to 0.

When split_transform_flag[x0][y0][trafoDepth] is not present, it is inferred as follows:

- If one or more of the following conditions are true, the value of split_transform_flag[x0][y0][trafoDepth] is inferred to be equal to 1:
 - log2TrafoSize is greater than MaxTbLog2SizeY.
 - IntraSplitFlag is equal to 1 and trafoDepth is equal to 0.
 - interSplitFlag is equal to 1.
- Otherwise, the value of split_transform_flag[x0][y0][trafoDepth] is inferred to be equal to 0.

cbf_luma[x0][y0][trafoDepth] equal to 1 specifies that the luma transform block contains one or more transform coefficient levels not equal to 0. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered transform block relative to the top-left luma sample of the picture. The array index trafoDepth specifies the

current subdivision level of a coding block into blocks for the purpose of transform coding. `trafoDepth` is equal to 0 for blocks that correspond to coding blocks.

When `cbf_luma[x0][y0][trafoDepth]` is not present, it is inferred to be equal to 1.

`cbf_cb[x0][y0][trafoDepth]` equal to 1 specifies that the Cb transform block contains one or more transform coefficient levels not equal to 0. The array indices `x0`, `y0` specify the top-left location (`x0`, `y0`) of the considered transform block. The array index `trafoDepth` specifies the current subdivision level of a coding block into blocks for the purpose of transform coding. `trafoDepth` is equal to 0 for blocks that correspond to coding blocks.

When `cbf_cb[x0][y0][trafoDepth]` is not present, it is inferred to be equal to 0.

`cbf_cr[x0][y0][trafoDepth]` equal to 1 specifies that the Cr transform block contains one or more transform coefficient levels not equal to 0. The array indices `x0`, `y0` specify the top-left location (`x0`, `y0`) of the considered transform block. The array index `trafoDepth` specifies the current subdivision level of a coding block into blocks for the purpose of transform coding. `trafoDepth` is equal to 0 for blocks that correspond to coding blocks.

When `cbf_cr[x0][y0][trafoDepth]` is not present, it is inferred to be equal to 0.

7.4.9.9 Motion vector difference semantics

`abs_mvd_greater0_flag[compIdx]` specifies whether the absolute value of a motion vector component difference is greater than 0.

`abs_mvd_greater1_flag[compIdx]` specifies whether the absolute value of a motion vector component difference is greater than 1.

When `abs_mvd_greater1_flag[compIdx]` is not present, it is inferred to be equal to 0.

`abs_mvd_minus2[compIdx]` plus 2 specifies the absolute value of a motion vector component difference.

When `abs_mvd_minus2[compIdx]` is not present, it is inferred to be equal to -1.

`mvd_sign_flag[compIdx]` specifies the sign of a motion vector component difference as follows:

- If `mvd_sign_flag[compIdx]` is equal to 0, the corresponding motion vector component difference has a positive value.
- Otherwise (`mvd_sign_flag[compIdx]` is equal to 1), the corresponding motion vector component difference has a negative value.

When `mvd_sign_flag[compIdx]` is not present, it is inferred to be equal to 0.

The motion vector difference `lMvd[compIdx]` for `compIdx = 0..1` is derived as follows:

$$\text{lMvd}[\text{compIdx}] = \text{abs_mvd_greater0_flag}[\text{compIdx}] * (\text{abs_mvd_minus2}[\text{compIdx}] + 2) * (1 - 2 * \text{mvd_sign_flag}[\text{compIdx}]) \quad (7-73)$$

The variable `MvdLX[x0][y0][compIdx]`, with `X` being 0 or 1, specifies the difference between a list `X` vector component to be used and its prediction. The value of `MvdLX[x0][y0][compIdx]` shall be in the range of -2^{15} to $2^{15} - 1$, inclusive. The array indices `x0`, `y0` specify the location (`x0`, `y0`) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture. The horizontal motion vector component difference is assigned `compIdx = 0` and the vertical motion vector component is assigned `compIdx = 1`.

- If `refList` is equal to 0, `MvdL0[x0][y0][compIdx]` is set equal to `lMvd[compIdx]` for `compIdx = 0..1`.
- Otherwise (`refList` is equal to 1), `MvdL1[x0][y0][compIdx]` is set equal to `lMvd[compIdx]` for `compIdx = 0..1`.

7.4.9.10 Transform unit semantics

The transform coefficient levels are represented by the arrays `TransCoeffLevel[x0][y0][cIdx][xC][yC]`, which are either specified in clause 7.3.8.11 or inferred as follows. The array indices `x0`, `y0` specify the location (`x0`, `y0`) of the top-left luma sample of the considered transform block relative to the top-left luma sample of the picture. The array index `cIdx` specifies an indicator for the colour component; it is equal to 0 for Y, 1 for Cb, and 2 for Cr. The array indices `xC` and `yC` specify the transform coefficient location (`xC`, `yC`) within the current transform block. When the value of `TransCoeffLevel[x0][y0][cIdx][xC][yC]` is not specified in clause 7.3.8.11, it is inferred to be equal to 0.

`tu_residual_act_flag[x0][y0]` equal to 1 specifies that adaptive colour transform is applied to the residual samples of the current transform unit. `tu_residual_act_flag[x0][y0]` equal to 0 specifies that adaptive colour transform is not applied to the residual samples of the current transform unit. When `tu_residual_act_flag[x0][y0]` is not present, it is inferred to be equal to 0.

When `cu_transquant_bypass_flag` is equal to 1 and `bit_depth_luma_minus8` is not equal to `bit_depth_chroma_minus8`, the value of `tu_residual_act_flag[x0][y0]`, when present, shall be equal to 0.

7.4.9.11 Residual coding semantics

For intra prediction, different scanning orders are used. The variable `scanIdx` specifies which scan order is used where `scanIdx` equal to 0 specifies an up-right diagonal scan order, `scanIdx` equal to 1 specifies a horizontal scan order, and `scanIdx` equal to 2 specifies a vertical scan order. The value of `scanIdx` is derived as follows:

- If `CuPredMode[x0][y0]` is equal to `MODE_INTRA` and one or more of the following conditions are true:
 - `log2TrafoSize` is equal to 2.
 - `log2TrafoSize` is equal to 3 and `cIdx` is equal to 0.
 - `log2TrafoSize` is equal to 3 and `ChromaArrayType` is equal to 3.

`predModeIntra` is derived as follows:

- If `cIdx` is equal to 0, `predModeIntra` is set equal to `IntraPredModeY[x0][y0]`.
- Otherwise, `predModeIntra` is set equal to `IntraPredModeC`.

`scanIdx` is derived as follows:

- If `predModeIntra` is in the range of 6 to 14, inclusive, `scanIdx` is set equal to 2.
 - Otherwise if `predModeIntra` is in the range of 22 to 30, inclusive, `scanIdx` is set equal to 1.
 - Otherwise, `scanIdx` is set equal to 0.
- Otherwise, `scanIdx` is set equal to 0.

`transform_skip_flag[x0][y0][cIdx]` specifies whether a transform is applied to the associated transform block or not: The array indices `x0`, `y0` specify the location (`x0`, `y0`) of the top-left luma sample of the considered transform block relative to the top-left luma sample of the picture. The array index `cIdx` specifies an indicator for the colour component; it is equal to 0 for luma, equal to 1 for Cb and equal to 2 for Cr. `transform_skip_flag[x0][y0][cIdx]` equal to 1 specifies that no transform is applied to the current transform block. `transform_skip_flag[x0][y0][cIdx]` equal to 0 specifies that the decision whether transform is applied to the current transform block or not depends on other syntax elements. When `transform_skip_flag[x0][y0][cIdx]` is not present, it is inferred to be equal to 0.

`explicit_rdpem_flag[x0][y0][cIdx]` specifies whether the residual modification process for blocks using a transform bypass is applied to the associated transform block or not. The array indices `x0`, `y0` specify the location (`x0`, `y0`) of the top-left luma sample of the considered transform block relative to the top-left luma sample of the picture. The array index `cIdx` specifies an indicator for the colour component; it is equal to 0 for luma, equal to 1 for Cb and equal to 2 for Cr. `explicit_rdpem_flag[x0][y0][cIdx]` equal to 1 specifies that the residual modification process is applied to the current transform block. `explicit_rdpem_flag[x0][y0][cIdx]` equal to 0 specifies that no residual modification process is applied to the current transform block. When `explicit_rdpem_flag[x0][y0][cIdx]` is not present, it is inferred to be equal to 0.

`explicit_rdpem_dir_flag[x0][y0][cIdx]` specifies the direction to be used by the residual modification process for the associated transform block. The array indices `x0`, `y0` specify the location (`x0`, `y0`) of the top-left luma sample of the considered transform block relative to the top-left luma sample of the picture. The array index `cIdx` specifies an indicator for the colour component; it is equal to 0 for luma, equal to 1 for Cb and equal to 2 for Cr.

`last_sig_coeff_x_prefix` specifies the prefix of the column position of the last significant coefficient in scanning order within a transform block. The values of `last_sig_coeff_x_prefix` shall be in the range of 0 to (`log2TrafoSize << 1`) – 1, inclusive.

`last_sig_coeff_y_prefix` specifies the prefix of the row position of the last significant coefficient in scanning order within a transform block. The values of `last_sig_coeff_y_prefix` shall be in the range of 0 to (`log2TrafoSize << 1`) – 1, inclusive.

`last_sig_coeff_x_suffix` specifies the suffix of the column position of the last significant coefficient in scanning order within a transform block. The values of `last_sig_coeff_x_suffix` shall be in the range of 0 to (1 << ((`last_sig_coeff_x_prefix` >> 1) – 1)) – 1, inclusive.

The column position of the last significant coefficient in scanning order within a transform block `LastSignificantCoeffX` is derived as follows:

- If `last_sig_coeff_x_suffix` is not present, the following applies:

$$\text{LastSignificantCoeffX} = \text{last_sig_coeff_x_prefix} \quad (7-74)$$

- Otherwise (last_sig_coeff_x_suffix is present), the following applies:

$$\text{LastSignificantCoeffX} = (1 \ll ((\text{last_sig_coeff_x_prefix} \gg 1) - 1)) * (2 + (\text{last_sig_coeff_x_prefix} \& 1)) + \text{last_sig_coeff_x_suffix} \quad (7-75)$$

last_sig_coeff_y_suffix specifies the suffix of the row position of the last significant coefficient in scanning order within a transform block. The values of last_sig_coeff_y_suffix shall be in the range of 0 to $(1 \ll ((\text{last_sig_coeff_y_prefix} \gg 1) - 1)) - 1$, inclusive.

The row position of the last significant coefficient in scanning order within a transform block LastSignificantCoeffY is derived as follows:

- If last_sig_coeff_y_suffix is not present, the following applies:

$$\text{LastSignificantCoeffY} = \text{last_sig_coeff_y_prefix} \quad (7-76)$$

- Otherwise (last_sig_coeff_y_suffix is present), the following applies:

$$\text{LastSignificantCoeffY} = (1 \ll ((\text{last_sig_coeff_y_prefix} \gg 1) - 1)) * (2 + (\text{last_sig_coeff_y_prefix} \& 1)) + \text{last_sig_coeff_y_suffix} \quad (7-77)$$

When scanIdx is equal to 2, the coordinates are swapped as follows:

$$(\text{LastSignificantCoeffX}, \text{LastSignificantCoeffY}) = \text{Swap}(\text{LastSignificantCoeffX}, \text{LastSignificantCoeffY}) \quad (7-78)$$

coded_sub_block_flag[xS][yS] specifies the following for the sub-block at location (xS, yS) within the current transform block, where a sub-block is a (4x4) array of 16 transform coefficient levels:

- If coded_sub_block_flag[xS][yS] is equal to 0, the 16 transform coefficient levels of the sub-block at location (xS, yS) are inferred to be equal to 0.
- Otherwise (coded_sub_block_flag[xS][yS] is equal to 1), the following applies:
 - If (xS, yS) is equal to (0, 0) and (LastSignificantCoeffX, LastSignificantCoeffY) is not equal to (0, 0), at least one of the 16 sig_coeff_flag syntax elements is present for the sub-block at location (xS, yS).
 - Otherwise, at least one of the 16 transform coefficient levels of the sub-block at location (xS, yS) has a non-zero value.

When coded_sub_block_flag[xS][yS] is not present, it is inferred as follows:

- If one or more of the following conditions are true, coded_sub_block_flag[xS][yS] is inferred to be equal to 1:
 - (xS, yS) is equal to (0, 0).
 - (xS, yS) is equal to (LastSignificantCoeffX >> 2, LastSignificantCoeffY >> 2).
- Otherwise, coded_sub_block_flag[xS][yS] is inferred to be equal to 0.

sig_coeff_flag[xC][yC] specifies for the transform coefficient location (xC, yC) within the current transform block whether the corresponding transform coefficient level at the location (xC, yC) is non-zero as follows:

- If sig_coeff_flag[xC][yC] is equal to 0, the transform coefficient level at the location (xC, yC) is set equal to 0.
- Otherwise (sig_coeff_flag[xC][yC] is equal to 1), the transform coefficient level at the location (xC, yC) has a non-zero value.

When sig_coeff_flag[xC][yC] is not present, it is inferred as follows:

- If (xC, yC) is the last significant location (LastSignificantCoeffX, LastSignificantCoeffY) in scan order or all of the following conditions are true, sig_coeff_flag[xC][yC] is inferred to be equal to 1:
 - (xC & 3, yC & 3) is equal to (0, 0).
 - inferSbDcSigCoeffFlag is equal to 1.
 - coded_sub_block_flag[xS][yS] is equal to 1.
- Otherwise, sig_coeff_flag[xC][yC] is inferred to be equal to 0.

coeff_abs_level_greater1_flag[n] specifies for the scanning position *n* whether there are absolute values of transform coefficient levels greater than 1.

When **coeff_abs_level_greater1_flag[n]** is not present, it is inferred to be equal to 0.

coeff_abs_level_greater2_flag[n] specifies for the scanning position *n* whether there are absolute values of transform coefficient levels greater than 2.

When **coeff_abs_level_greater2_flag[n]** is not present, it is inferred to be equal to 0.

coeff_sign_flag[n] specifies the sign of a transform coefficient level for the scanning position *n* as follows:

- If **coeff_sign_flag[n]** is equal to 0, the corresponding transform coefficient level has a positive value.
- Otherwise (**coeff_sign_flag[n]** is equal to 1), the corresponding transform coefficient level has a negative value.

When **coeff_sign_flag[n]** is not present, it is inferred to be equal to 0.

coeff_abs_level_remaining[n] is the remaining absolute value of a transform coefficient level that is coded with Golomb-Rice code at the scanning position *n*. When **coeff_abs_level_remaining[n]** is not present, it is inferred to be equal to 0.

It is a requirement of bitstream conformance that the value of **coeff_abs_level_remaining[n]** shall be constrained such that the corresponding value of **TransCoeffLevel[x0][y0][cIdx][xC][yC]** is in the range of **CoeffMin_Y** to **CoeffMax_Y**, inclusive, for **cIdx** equal to 0 and in the range of **CoeffMin_C** to **CoeffMax_C**, inclusive, for **cIdx** not equal to 0.

7.4.9.12 Cross-component prediction semantics

log2_res_scale_abs_plus1[c] minus 1 specifies the base 2 logarithm of the magnitude of the scaling factor **ResScaleVal** used in cross-component residual prediction. When not present, **log2_res_scale_abs_plus1** is inferred to be equal to 0.

res_scale_sign_flag[c] specifies the sign of the scaling factor used in cross-component residual prediction as follows:

- If **res_scale_sign_flag[c]** is equal to 0, the corresponding **ResScaleVal** has a positive value.
- Otherwise (**res_scale_sign_flag[c]** is equal to 1), the corresponding **ResScaleVal** has a negative value.

The variable **ResScaleVal[cIdx][x0][y0]** specifies the scaling factor used in cross-component residual prediction. The array indices *x0*, *y0* specify the location (*x0*, *y0*) of the top-left luma sample of the considered transform block relative to the top-left luma sample of the picture. The array index *cIdx* specifies an indicator for the colour component; it is equal to 1 for Cb and equal to 2 for Cr.

The variable **ResScaleVal[cIdx][x0][y0]** is derived as follows:

- If **log2_res_scale_abs_plus1[cIdx – 1]** is equal to 0, the following applies:

$$\text{ResScaleVal}[\text{cIdx}][\text{x0}][\text{y0}] = 0 \quad (7-79)$$

- Otherwise (**log2_res_scale_abs_plus1[cIdx – 1]** is not equal to 0), the following applies:

$$\text{ResScaleVal}[\text{cIdx}][\text{x0}][\text{y0}] = (1 \ll (\text{log2_res_scale_abs_plus1}[\text{cIdx} - 1] - 1)) * (1 - 2 * \text{res_scale_sign_flag}[\text{cIdx} - 1]) \quad (7-80)$$

7.4.9.13 Palette semantics

In the following semantics, the array indices *x0*, *y0* specify the location (*x0*, *y0*) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

The predictor palette consists of palette entries from previous coding units that are used to predict the entries in the current palette.

The variable **PredictorPaletteSize** specifies the size of the predictor palette. **PredictorPaletteSize** is derived as specified in clause 8.4.4.2.7.

The variable **PalettePredictorEntryReuseFlags[i]** equal to 1 specifies that the *i*-th entry in the predictor palette is reused in the current palette. **PalettePredictorEntryReuseFlags[i]** equal to 0 specifies that the *i*-th entry in the predictor palette is not an entry in the current palette. All elements of the array **PalettePredictorEntryReuseFlags[i]** are initialized to 0.

palette_predictor_run is used to determine the number of zeros that precede a non-zero entry in the array **PalettePredictorEntryReuseFlags**.

It is a requirement of bitstream conformance that the value of **palette_predictor_run** shall be in the range of 0 to (**PredictorPaletteSize** – **predictorEntryIdx**), inclusive, where **predictorEntryIdx** corresponds to the current position in the

array `PalettePredictorEntryReuseFlags`. The variable `NumPredictedPaletteEntries` specifies the number of entries in the current palette that are reused from the predictor palette. The value of `NumPredictedPaletteEntries` shall be in the range of 0 to `palette_max_size`, inclusive.

num_signalled_palette_entries specifies the number of entries in the current palette that are explicitly signalled.

When `num_signalled_palette_entries` is not present, it is inferred to be equal to 0.

The variable `CurrentPaletteSize` specifies the size of the current palette and is derived as follows:

$$\text{CurrentPaletteSize} = \text{NumPredictedPaletteEntries} + \text{num_signalled_palette_entries} \quad (7-81)$$

The value of `CurrentPaletteSize` shall be in the range of 0 to `palette_max_size`, inclusive.

new_palette_entries[cIdx][i] specifies the value for the *i*-th signalled palette entry for the colour component *cIdx*.

The variable `PredictorPaletteEntries[cIdx][i]` specifies the *i*-th element in the predictor palette for the colour component *cIdx*.

The variable `CurrentPaletteEntries[cIdx][i]` specifies the *i*-th element in the current palette for the colour component *cIdx* and is derived as follows:

```

numComps = ( ChromaArrayType == 0 ) ? 1 : 3
numPredictedPaletteEntries = 0
for( i = 0; i < PredictorPaletteSize; i++ )
    if( PalettePredictorEntryReuseFlags[ i ] ) {
        for( cIdx = 0; cIdx < numComps; cIdx++ )
            CurrentPaletteEntries[ cIdx ][ numPredictedPaletteEntries ] =
PredictorPaletteEntries[ cIdx ][ i ]
            numPredictedPaletteEntries++
    }
for( cIdx = 0; cIdx < numComps; cIdx++ )
    for( i = 0; i < num_signalled_palette_entries; i++ )
        CurrentPaletteEntries[ cIdx ][ numPredictedPaletteEntries + i ] = new_palette_entries[ cIdx ][ i ]

```

(7-82)

palette_escape_val_present_flag equal to 1 specifies that the current coding unit contains at least one escape-coded sample. `palette_escape_val_present_flag` equal to 0 specifies that there are no escape-coded samples in the current coding unit. When not present, the value of `palette_escape_val_present_flag` is inferred to be equal to 1.

The variable `MaxPaletteIndex` specifies the maximum possible value for a palette index for the current coding unit. The value of `MaxPaletteIndex` is set equal to `CurrentPaletteSize - 1 + palette_escape_val_present_flag`.

num_palette_indices_minus1 plus 1 is the number of palette indices explicitly signalled or inferred for the current block.

When `num_palette_indices_minus1` is not present, it is inferred to be equal to 0.

palette_idx_idc is an indication of an index to the array represented by `CurrentPaletteEntries`. The value of `palette_idx_idc` shall be in the range of 0 to `MaxPaletteIndex`, inclusive, for the first index in the block and in the range of 0 to (`MaxPaletteIndex - 1`), inclusive, for the remaining indices in the block.

When `palette_idx_idc` is not present, it is inferred to be equal to 0.

The variable `PaletteIndexIdc[i]` stores the *i*-th `palette_idx_idc` explicitly signalled or inferred. All elements of the array `PaletteIndexIdc[i]` are initialized to 0.

copy_above_indices_for_final_run_flag equal to 1 specifies that the palette indices of the last positions in the coding unit are copied from the palette indices in the row above. **copy_above_indices_for_final_run_flag** equal to 0 specifies that the palette indices of the last positions in the coding unit are copied from `PaletteIndexIdc[num_palette_indices_minus1]`.

When **copy_above_indices_for_final_run_flag** is not present, it is inferred to be equal to 0.

palette_transpose_flag equal to 1 specifies that the transpose process is applied to the reconstructed sample array at the output of palette mode decoding as specified in clause 8.4.4.2.7 in the current coding unit. **palette_transpose_flag** equal to 0 specifies that the transpose process is not applied to the reconstructed sample array at the output of palette mode decoding clause 8.4.4.2.7 in the current coding unit. When not present, the value of **palette_transpose_flag** is inferred to be equal to 0.

copy_above_palette_indices_flag equal to 1 specifies that the palette index is equal to the palette index at the same location in the row above. **copy_above_palette_indices_flag** equal to 0 specifies that an indication of the palette index of the sample is coded in the bitstream or inferred.

The variable `CopyAboveIndicesFlag[xC][yC]` equal to 1 specifies that the palette index is copied from the palette index in the row above. `CopyAboveIndicesFlag[xC][yC]` equal to 0 specifies that the palette index is explicitly coded in the bitstream or inferred. The array indices `xC`, `yC` specify the location (`xC`, `yC`) of the sample relative to the top-left luma sample of the picture.

The variable `PaletteIndexMap[xC][yC]` specifies a palette index, which is an index to the array represented by `CurrentPaletteEntries`. The array indices `xC`, `yC` specify the location (`xC`, `yC`) of the sample relative to the top-left luma sample of the picture. The value of `PaletteIndexMap[xC][yC]` shall be in the range of 0 to `MaxPaletteIndex`, inclusive.

The variable `adjustedRefPaletteIndex` is derived as follows:

```
adjustedRefPaletteIndex = MaxPaletteIndex + 1
log2BlkSize = Log2( nCbs ) - 2
if( PaletteScanPos > 0 ) {
    xcPrev = x0 + ScanOrder[ log2BlkSize ][ 3 ][ PaletteScanPos - 1 ][ 0 ]
    ycPrev = y0 + ScanOrder[ log2BlkSize ][ 3 ][ PaletteScanPos - 1 ][ 1 ]
    if( CopyAboveIndicesFlag[ xcPrev ][ ycPrev ] == 0 )
        adjustedRefPaletteIndex = PaletteIndexMap[ xcPrev ][ ycPrev ]
    else
        adjustedRefPaletteIndex = PaletteIndexMap[ xC ][ yC - 1 ]
}
```

(7-83)

When `CopyAboveIndicesFlag[xC][yC]` is equal to 0, the variable `CurrPaletteIndex` is derived as follows:

```
if( CurrPaletteIndex >= adjustedRefPaletteIndex )
    CurrPaletteIndex++
```

(7-84)

palette_run_prefix, when present, is used in the derivation of the variable `PaletteRunMinus1`.

palette_run_suffix is used in the derivation of the variable `PaletteRunMinus1`. When not present, the value of **palette_run_suffix** is inferred to be equal to 0.

When `RunToEnd` is equal to 0, the variable `PaletteRunMinus1` is derived as follows:

- If `PaletteMaxRunMinus1` is equal to 0, `PaletteRunMinus1` is set equal to 0.
- Otherwise (`PaletteMaxRunMinus1` is greater than 0) the following applies:
 - If **palette_run_prefix** is less than 2, the following applies:

```
PaletteRunMinus1 = palette_run_prefix
```

(7-85)

- Otherwise (palette_run_prefix is greater than or equal to 2), the following applies:

$$\begin{aligned}\text{PrefixOffset} &= 1 \ll (\text{palette_run_prefix} - 1) \\ \text{PaletteRunMinus1} &= \text{PrefixOffset} + \text{palette_run_suffix}\end{aligned}\quad (7-86)$$

The variable PaletteRunMinus1 is used as follows:

- If CopyAboveIndicesFlag[xC][yC] is equal to 0, PaletteRunMinus1 specifies the number of consecutive locations minus 1 with the same palette index.
- Otherwise, PaletteRunMinus1 specifies the number of consecutive locations minus 1 with the same palette index as used in the corresponding position in the row above.

When RunToEnd is equal to 0, the variable PaletteMaxRunMinus1 represents the maximum possible value for PaletteRunMinus1 and it is a requirement of bitstream conformance that the value of PaletteMaxRunMinus1 shall be greater than or equal to 0.

palette_escape_val specifies the quantized escape-coded sample value for a component.

The variable PaletteEscapeVal[cIdx][xC][yC] specifies the escape value of a sample for which PaletteIndexMap[xC][yC] is equal to MaxPaletteIndex and palette_escape_val_present_flag is equal to 1. The array index cIdx specifies the colour component. The array indices xC, yC specify the location (xC, yC) of the sample relative to the top-left luma sample of the picture.

It is a requirement of bitstream conformance that PaletteEscapeVal[cIdx][xC][yC] shall be in the range of 0 to $(1 \ll (\text{BitDepth}_Y + 1)) - 1$, inclusive, for cIdx equal to 0, and in the range of 0 to $(1 \ll (\text{BitDepth}_C + 1)) - 1$, inclusive, for cIdx not equal to 0.

7.4.9.14 Delta QP semantics

cu_qp_delta_abs, when present, specifies the absolute value of the difference CuQpDeltaVal between the luma quantization parameter of the current coding unit and its prediction.

cu_qp_delta_sign_flag, when present, specifies the sign of the difference CuQpDeltaVal between the luma quantization parameter of the current coding unit and its prediction.

It is a requirement of bitstream conformance that the value of CuQpDeltaVal shall be in the range of $-(26 + \text{QpBdOffsetY} / 2)$ to $+(25 + \text{QpBdOffsetY} / 2)$, inclusive.

7.4.9.15 Chroma QP offset semantics

cu_chroma_qp_offset_flag, when present and equal to 1, specifies that an entry in the cb_qp_offset_list[] is used to determine the value of CuQpOffsetCb and a corresponding entry in the cr_qp_offset_list[] is used to determine the value of CuQpOffsetCr. cu_chroma_qp_offset_flag equal to 0 specifies that these lists are not used to determine the values of CuQpOffsetCb and CuQpOffsetCr.

cu_chroma_qp_offset_idx, when present, specifies the index into the cb_qp_offset_list[] and cr_qp_offset_list[] that is used to determine the value of CuQpOffsetCb and CuQpOffsetCr. When present, the value of cu_chroma_qp_offset_idx shall be in the range of 0 to chroma_qp_offset_list_len_minus1, inclusive. When not present, the value of cu_chroma_qp_offset_idx is inferred to be equal to 0.

When cu_chroma_qp_offset_flag is present, the following applies:

- The variable IsCuChromaQpOffsetCoded is set equal to 1.
- The variables CuQpOffsetCb and CuQpOffsetCr are derived as follows:
 - If cu_chroma_qp_offset_flag is equal to 1, the following applies:

$$\text{CuQpOffsetCb} = \text{cb_qp_offset_list}[\text{cu_chroma_qp_offset_idx}] \quad (7-87)$$

$$\text{CuQpOffsetCr} = \text{cr_qp_offset_list}[\text{cu_chroma_qp_offset_idx}] \quad (7-88)$$

- Otherwise (cu_chroma_qp_offset_flag is equal to 0), CuQpOffsetCb and CuQpOffsetCr are both set equal to 0.

NOTE – When cu_chroma_qp_offset_enabled_flag is equal to 0, CuQpOffsetCb and CuQpOffsetCr are not modified after being initialized to 0 for the slice as specified in clause 7.4.7.1.

8 Decoding process

8.1 General decoding process

8.1.1 General

Input to this process is a bitstream. Output of this process is a list of decoded pictures.

The decoding process is specified such that all decoders that conform to a specified profile, tier and level will produce numerically identical cropped decoded output pictures when invoking the decoding process associated with that profile for a bitstream conforming to that profile, tier and level. Any decoding process that produces identical cropped decoded output pictures to those produced by the process described herein (with the correct output order or output timing, as specified) conforms to the decoding process requirements of this Specification.

NOTE 1 – For the purpose of best-effort decoding, a decoder that conforms to a particular profile at a given tier and level may additionally decode some bitstreams conforming to a different tier, level or profile without necessarily using a decoding process that produces numerically identical cropped decoded output pictures to those produced by the process specified herein (without claiming conformance to the other profile, tier and level).

At the beginning of decoding a coded video sequence group (CVSG), after activating the VPS RBSP that is active for the entire CVSG and before decoding any VCL NAL units of the CVSG, clause 8.1.2 is invoked with the CVSG as input.

8.1.2 CVSG decoding process

Input to this process is a CVSG. Output of this process is a list of decoded pictures.

The layer identifier list TargetDecLayerIdList, which specifies the list of nuh_layer_id values, in increasing order of nuh_layer_id values, of the NAL units to be decoded, is specified as follows:

- If some external means, not specified in this Specification, is available to set TargetDecLayerIdList, TargetDecLayerIdList is set by the external means.
- Otherwise, if the decoding process is invoked in a bitstream conformance test as specified in clause C.1, TargetDecLayerIdList is set as specified in clause C.1.
- Otherwise, TargetDecLayerIdList contains only one nuh_layer_id value that is equal to 0.

The variable HighestTid, which identifies the highest temporal sub-layer to be decoded, is specified as follows:

- If some external means, not specified in this Specification, is available to set HighestTid, HighestTid is set by the external means.
- Otherwise, if the decoding process is invoked in a bitstream conformance test as specified in clause C.1, HighestTid is set as specified in clause C.1.
- Otherwise, HighestTid is set equal to sps_max_sub_layers_minus1.

The variable SubPicHrdFlag is specified as follows:

- If the decoding process is invoked in a bitstream conformance test as specified in clause C.1, SubPicHrdFlag is set as specified in clause C.1.
- Otherwise, SubPicHrdFlag is set equal to (SubPicHrdPreferredFlag && sub_pic_hrd_params_present_flag).

The sub-bitstream extraction process as specified in clause 10 is applied with the CVSG, HighestTid and TargetDecLayerIdList as inputs, and the output is assigned to a bitstream referred to as BitstreamToDecode.

Clause 8.1.3 is repeatedly invoked for each coded picture with nuh_layer_id equal to 0 in BitstreamToDecode in decoding order.

8.1.3 Decoding process for a coded picture with nuh_layer_id equal to 0

The decoding processes specified in this clause apply to each coded picture with nuh_layer_id equal to 0, referred to as the current picture and denoted by the variable CurrPic, in BitstreamToDecode.

Depending on the value of chroma_format_idc, the number of sample arrays of the current picture is as follows:

- If chroma_format_idc is equal to 0, the current picture consists of 1 sample array S_L.
- Otherwise (chroma_format_idc is not equal to 0), the current picture consists of 3 sample arrays S_L, S_{Cb}, S_{Cr}.

The decoding process for the current picture takes as inputs the syntax elements and upper-case variables from clause 7. When interpreting the semantics of each syntax element in each NAL unit, the term "the bitstream" (or part thereof, e.g., a CVS of the bitstream) refers to BitstreamToDecode (or part thereof).

When the current picture is a BLA picture that has nal_unit_type equal to BLA_W_LP or is a CRA picture, the following applies:

- If some external means not specified in this Specification is available to set the variable UseAltCpbParamsFlag to a value, UseAltCpbParamsFlag is set equal to the value provided by the external means.
- Otherwise, the value of UseAltCpbParamsFlag is set equal to 0.

When the current picture is an IRAP picture, the following applies:

- If the current picture is an IDR picture, a BLA picture, the first picture in the bitstream in decoding order, or the first picture that follows an end of sequence NAL unit in decoding order, the variable NoRaslOutputFlag is set equal to 1.
- Otherwise, if some external means not specified in this Specification is available to set the variable HandleCraAsBlaFlag to a value for the current picture, the variable HandleCraAsBlaFlag is set equal to the value provided by the external means and the variable NoRaslOutputFlag is set equal to HandleCraAsBlaFlag.
- Otherwise, the variable HandleCraAsBlaFlag is set equal to 0 and the variable NoRaslOutputFlag is set equal to 0.

Depending on the value of separate_colour_plane_flag, the decoding process is structured as follows:

- If separate_colour_plane_flag is equal to 0, the decoding process is invoked a single time with the current picture being the output.
- Otherwise (separate_colour_plane_flag is equal to 1), the decoding process is invoked three times. Inputs to the decoding process are all NAL units of the coded picture with identical value of colour_plane_id. The decoding process of NAL units with a particular value of colour_plane_id is specified as if only a CVS with monochrome colour format with that particular value of colour_plane_id would be present in the bitstream. The output of each of the three decoding processes is assigned to one of the 3 sample arrays of the current picture, with the NAL units with colour_plane_id equal to 0, 1 and 2 being assigned to S_L, S_{Cb} and S_{Cr}, respectively.

NOTE 1 – The variable ChromaArrayType is derived as equal to 0 when separate_colour_plane_flag is equal to 1 and chroma_format_idc is equal to 3. In the decoding process, the value of this variable is evaluated resulting in operations identical to that of monochrome pictures (when chroma_format_idc is equal to 0).

The decoding process operates as follows for the current picture CurrPic:

1. The decoding of NAL units is specified in clause 8.2.
2. The processes in clause 8.3 specify the following decoding processes using syntax elements in the slice segment layer and above:
 - Variables and functions relating to picture order count are derived as specified in clause 8.3.1. This needs to be invoked only for the first slice segment of a picture.
 - The decoding process for RPS in clause 8.3.2 is invoked, wherein reference pictures may be marked as "unused for reference" or "used for long-term reference". This needs to be invoked only for the first slice segment of a picture.
 - A picture storage buffer in the DPB is allocated for storage of the decoded sample values of the current picture after the invocation of the in-loop filter process as specified in clause 8.7. This version of the current decoded picture is referred to as the current decoded picture after the invocation of the in-loop filter process. When TwoVersionsOfCurrDecPicFlag is equal to 0 and pps_curr_pic_ref_enabled_flag is equal to 1, this picture storage buffer is marked as "used for long-term reference". When TwoVersionsOfCurrDecPicFlag is equal to 1, another picture storage buffer in the DPB is allocated for storage of the decoded sample values of the current picture immediately before the invocation of the in-loop filter process as specified in clause 8.7, and is marked as "used for long-term reference". This version of the current decoded picture is referred to as the current decoded picture before the invocation of the in-loop filter process. This needs to be invoked only for the first slice segment of a picture.

NOTE 2 – When TwoVersionsOfCurrDecPicFlag is equal to 0, there is only one version of the current decoded picture. In this case, if pps_curr_pic_ref_enabled_flag is equal to 1, the current decoded picture is marked as "used for long-term reference" during the decoding of the current picture and will be marked as "used for short-term reference" at the end of the decoding of the current picture, otherwise it is not marked at all during the decoding of the current picture and will be marked as "used for short-term reference" at the end of the decoding of the current picture. When TwoVersionsOfCurrDecPicFlag is equal to 1, there are two versions of the current decoded picture, one of which is marked as "used for long-term reference" during the decoding of the current picture and will be marked as "unused for reference" at the end of the decoding of the current picture, and the

other version is not marked at all during the decoding of the current picture and will be marked as "used for short-term reference" at the end of the decoding of the current picture.

- When the current picture is a BLA picture or is a CRA picture with NoRaslOutputFlag equal to 1, the decoding process for generating unavailable reference pictures specified in clause 8.3.3 is invoked, which needs to be invoked only for the first slice segment of a picture.
 - PicOutputFlag is set as follows:
 - If the current picture is a RASL picture and NoRaslOutputFlag of the associated IRAP picture is equal to 1, PicOutputFlag is set equal to 0.
 - Otherwise, PicOutputFlag is set equal to pic_output_flag.
 - At the beginning of the decoding process for each P or B slice, the decoding process for reference picture lists construction specified in clause 8.3.4 is invoked for derivation of reference picture list 0 (RefPicList0) and, when decoding a B slice, reference picture list 1 (RefPicList1), and the decoding process for collocated picture and no backward prediction flag specified in clause 8.3.5 is invoked for derivation of the variables ColPic and NoBackwardPredFlag.
3. The processes in clauses 8.4, 8.5, 8.6 and 8.7 specify decoding processes using syntax elements in all syntax structure layers. It is a requirement of bitstream conformance that the coded slices of the picture shall contain slice segment data for every CTU of the picture, such that the division of the picture into slices, the division of the slices into slice segments and the division of the slice segments into CTUs each forms a partitioning of the picture.
 4. After all slices of the current picture have been decoded, the current decoded picture after the invocation of the in-loop filter process as specified in clause 8.7 is marked as "used for short-term reference". When TwoVersionsOfCurrDecPicFlag is equal to 1, the current decoded picture before the invocation of the in-loop filter process as specified in clause 8.7 is marked as "unused for reference".

8.2 NAL unit decoding process

Inputs to this process are NAL units of the current picture and their associated non-VCL NAL units.

Outputs of this process are the parsed RBSP syntax structures encapsulated within the NAL units.

The decoding process for each NAL unit extracts the RBSP syntax structure from the NAL unit and then parses the RBSP syntax structure.

8.3 Slice decoding process

8.3.1 Decoding process for picture order count

Output of this process is PicOrderCntVal, the picture order count of the current picture.

Picture order counts are used to identify pictures, for deriving motion parameters in merge mode and motion vector prediction, and for decoder conformance checking (see clause C.5).

Each coded picture is associated with a picture order count variable, denoted as PicOrderCntVal.

When the current picture is not an IRAP picture with NoRaslOutputFlag equal to 1, the variables prevPicOrderCntLsb and prevPicOrderCntMsb are derived as follows:

- Let prevTid0Pic be the previous picture in decoding order that has TemporalId equal to 0 and that is not a RASL, RADL or SLNR picture.
- The variable prevPicOrderCntLsb is set equal to slice_pic_order_cnt_lsb of prevTid0Pic.
- The variable prevPicOrderCntMsb is set equal to PicOrderCntMsb of prevTid0Pic.

The variable PicOrderCntMsb of the current picture is derived as follows:

- If the current picture is an IRAP picture with NoRaslOutputFlag equal to 1, PicOrderCntMsb is set equal to 0.
- Otherwise, PicOrderCntMsb is derived as follows:

$$\begin{aligned}
 & \text{if} ((\text{slice_pic_order_cnt_lsb} < \text{prevPicOrderCntLsb}) \ \&\& \\
 & \quad ((\text{prevPicOrderCntLsb} - \text{slice_pic_order_cnt_lsb}) \geq (\text{MaxPicOrderCntLsb} / 2))) \\
 & \quad \text{PicOrderCntMsb} = \text{prevPicOrderCntMsb} + \text{MaxPicOrderCntLsb} \\
 & \text{else if} ((\text{slice_pic_order_cnt_lsb} > \text{prevPicOrderCntLsb}) \ \&\& \\
 & \quad ((\text{slice_pic_order_cnt_lsb} - \text{prevPicOrderCntLsb}) > (\text{MaxPicOrderCntLsb} / 2)))
 \end{aligned} \tag{8-1}$$

```

    PicOrderCntMsb = prevPicOrderCntMsb – MaxPicOrderCntLsb
else
    PicOrderCntMsb = prevPicOrderCntMsb

```

PicOrderCntVal is derived as follows:

$$\text{PicOrderCntVal} = \text{PicOrderCntMsb} + \text{slice_pic_order_cnt_lsb} \quad (8-2)$$

NOTE 1 – All IDR pictures will have PicOrderCntVal equal to 0 since slice_pic_order_cnt_lsb is inferred to be 0 for IDR pictures and prevPicOrderCntLsb and prevPicOrderCntMsb are both set equal to 0.

The value of PicOrderCntVal shall be in the range of -2^{31} to $2^{31} - 1$, inclusive. In one CVS, the PicOrderCntVal values for any two coded pictures shall not be the same.

The function PicOrderCnt(picX) is specified as follows:

$$\text{PicOrderCnt}(\text{picX}) = \text{PicOrderCntVal of the picture picX} \quad (8-3)$$

The function DiffPicOrderCnt(picA, picB) is specified as follows:

$$\text{DiffPicOrderCnt}(\text{picA}, \text{picB}) = \text{PicOrderCnt}(\text{picA}) - \text{PicOrderCnt}(\text{picB}) \quad (8-4)$$

The bitstream shall not contain data that result in values of DiffPicOrderCnt(picA, picB) used in the decoding process that are not in the range of -2^{15} to $2^{15} - 1$, inclusive.

NOTE 2 – Let X be the current picture and Y and Z be two other pictures in the same CVS, Y and Z are considered to be in the same output order direction from X when both DiffPicOrderCnt(X, Y) and DiffPicOrderCnt(X, Z) are positive or both are negative.

8.3.2 Decoding process for reference picture set

This process is invoked once per picture, after decoding of a slice header but prior to the decoding of any coding unit and prior to the decoding process for reference picture list construction for the slice as specified in clause 8.3.4. This process may result in one or more reference pictures in the DPB being marked as "unused for reference" or "used for long-term reference".

NOTE 1 – The RPS is an absolute description of the reference pictures used in the decoding process of the current and future coded pictures. The RPS signalling is explicit in the sense that all reference pictures included in the RPS are listed explicitly.

A decoded picture in the DPB can be marked as "unused for reference", "used for short-term reference" or "used for long-term reference", but only one among these three at any given moment during the operation of the decoding process. Assigning one of these markings to a picture implicitly removes another of these markings when applicable. When a picture is referred to as being marked as "used for reference", this collectively refers to the picture being marked as "used for short-term reference" or "used for long-term reference" (but not both).

The variable currPicLayerId is set equal to nuh_layer_id of the current picture.

When the current picture is an IRAP picture with NoRasOutputFlag equal to 1, all reference pictures with nuh_layer_id equal to currPicLayerId currently in the DPB (if any) are marked as "unused for reference".

Short-term reference pictures are identified by their PicOrderCntVal values. Long-term reference pictures are identified either by their PicOrderCntVal values or their slice_pic_order_cnt_lsb values.

Five lists of picture order count values are constructed to derive the RPS. These five lists are PocStCurrBefore, PocStCurrAfter, PocStFoll, PocLtCurr and PocLtFoll, with NumPocStCurrBefore, NumPocStCurrAfter, NumPocStFoll, NumPocLtCurr and NumPocLtFoll number of elements, respectively. The five lists and the five variables are derived as follows:

- If the current picture is an IDR picture, PocStCurrBefore, PocStCurrAfter, PocStFoll, PocLtCurr and PocLtFoll are all set to be empty, and NumPocStCurrBefore, NumPocStCurrAfter, NumPocStFoll, NumPocLtCurr and NumPocLtFoll are all set equal to 0.
- Otherwise, the following applies:

```

for( i = 0, j = 0, k = 0; i < NumNegativePics[ CurrRpsIdx ]; i++ )
    if( UsedByCurrPicS0[ CurrRpsIdx ][ i ] )
        PocStCurrBefore[ j++ ] = PicOrderCntVal + DeltaPocS0[ CurrRpsIdx ][ i ]
    else
        PocStFoll[ k++ ] = PicOrderCntVal + DeltaPocS0[ CurrRpsIdx ][ i ]
NumPocStCurrBefore = j

```

```

for( i = 0, j = 0; i < NumPositivePics[ CurrRpsIdx ]; i++ )
    if( UsedByCurrPicS1[ CurrRpsIdx ][ i ] )
        PocStCurrAfter[ j++ ] = PicOrderCntVal + DeltaPocS1[ CurrRpsIdx ][ i ]
    else
        PocStFoll[ k++ ] = PicOrderCntVal + DeltaPocS1[ CurrRpsIdx ][ i ]
NumPocStCurrAfter = j
NumPocStFoll = k
for( i = 0, j = 0, k = 0; i < num_long_term_sps + num_long_term_pics; i++ ) {
    pocLt = PocLsbLt[ i ]
    if( delta_poc_msb_present_flag[ i ] )
        pocLt += PicOrderCntVal - DeltaPocMsbCycleLt[ i ] * MaxPicOrderCntLsb -
                ( PicOrderCntVal & ( MaxPicOrderCntLsb - 1 ) )
    if( UsedByCurrPicLt[ i ] ) {
        PocLtCurr[ j ] = pocLt
        CurrDeltaPocMsbPresentFlag[ j++ ] = delta_poc_msb_present_flag[ i ]
    } else {
        PocLtFoll[ k ] = pocLt
        FollDeltaPocMsbPresentFlag[ k++ ] = delta_poc_msb_present_flag[ i ]
    }
}
NumPocLtCurr = j
NumPocLtFoll = k

```

(8-5)

where PicOrderCntVal is the picture order count of the current picture as specified in clause 8.3.1.

NOTE 2 – A value of CurrRpsIdx in the range of 0 to num_short_term_ref_pic_sets – 1, inclusive, indicates that a candidate short-term RPS from the active SPS for the current layer is being used, where CurrRpsIdx is the index of the candidate short-term RPS into the list of candidate short-term RPSs signalled in the active SPS for the current layer. CurrRpsIdx equal to num_short_term_ref_pic_sets indicates that the short-term RPS of the current picture is directly signalled in the slice header.

For each i in the range of 0 to NumPocLtCurr – 1, inclusive, when CurrDeltaPocMsbPresentFlag[i] is equal to 1, it is a requirement of bitstream conformance that the following conditions apply:

- There shall be no j in the range of 0 to NumPocStCurrBefore – 1, inclusive, for which PocLtCurr[i] is equal to PocStCurrBefore[j].
- There shall be no j in the range of 0 to NumPocStCurrAfter – 1, inclusive, for which PocLtCurr[i] is equal to PocStCurrAfter[j].
- There shall be no j in the range of 0 to NumPocStFoll – 1, inclusive, for which PocLtCurr[i] is equal to PocStFoll[j].
- There shall be no j in the range of 0 to NumPocLtCurr – 1, inclusive, where j is not equal to i , for which PocLtCurr[i] is equal to PocLtCurr[j].

For each i in the range of 0 to NumPocLtFoll – 1, inclusive, when FollDeltaPocMsbPresentFlag[i] is equal to 1, it is a requirement of bitstream conformance that the following conditions apply:

- There shall be no j in the range of 0 to NumPocStCurrBefore – 1, inclusive, for which PocLtFoll[i] is equal to PocStCurrBefore[j].
- There shall be no j in the range of 0 to NumPocStCurrAfter – 1, inclusive, for which PocLtFoll[i] is equal to PocStCurrAfter[j].
- There shall be no j in the range of 0 to NumPocStFoll – 1, inclusive, for which PocLtFoll[i] is equal to PocStFoll[j].
- There shall be no j in the range of 0 to NumPocLtFoll – 1, inclusive, where j is not equal to i , for which PocLtFoll[i] is equal to PocLtFoll[j].
- There shall be no j in the range of 0 to NumPocLtCurr – 1, inclusive, for which PocLtFoll[i] is equal to PocLtCurr[j].

For each i in the range of 0 to NumPocLtCurr – 1, inclusive, when CurrDeltaPocMsbPresentFlag[i] is equal to 0, it is a requirement of bitstream conformance that the following conditions apply:

- There shall be no j in the range of 0 to $\text{NumPocStCurrBefore} - 1$, inclusive, for which $\text{PocLtCurr}[i]$ is equal to $(\text{PocStCurrBefore}[j] \& (\text{MaxPicOrderCntLsb} - 1))$.
- There shall be no j in the range of 0 to $\text{NumPocStCurrAfter} - 1$, inclusive, for which $\text{PocLtCurr}[i]$ is equal to $(\text{PocStCurrAfter}[j] \& (\text{MaxPicOrderCntLsb} - 1))$.
- There shall be no j in the range of 0 to $\text{NumPocStFoll} - 1$, inclusive, for which $\text{PocLtCurr}[i]$ is equal to $(\text{PocStFoll}[j] \& (\text{MaxPicOrderCntLsb} - 1))$.
- There shall be no j in the range of 0 to $\text{NumPocLtCurr} - 1$, inclusive, where j is not equal to i , for which $\text{PocLtCurr}[i]$ is equal to $(\text{PocLtCurr}[j] \& (\text{MaxPicOrderCntLsb} - 1))$.

For each i in the range of 0 to $\text{NumPocLtFoll} - 1$, inclusive, when $\text{FollDeltaPocMsbPresentFlag}[i]$ is equal to 0, it is a requirement of bitstream conformance that the following conditions apply:

- There shall be no j in the range of 0 to $\text{NumPocStCurrBefore} - 1$, inclusive, for which $\text{PocLtFoll}[i]$ is equal to $(\text{PocStCurrBefore}[j] \& (\text{MaxPicOrderCntLsb} - 1))$.
- There shall be no j in the range of 0 to $\text{NumPocStCurrAfter} - 1$, inclusive, for which $\text{PocLtFoll}[i]$ is equal to $(\text{PocStCurrAfter}[j] \& (\text{MaxPicOrderCntLsb} - 1))$.
- There shall be no j in the range of 0 to $\text{NumPocStFoll} - 1$, inclusive, for which $\text{PocLtFoll}[i]$ is equal to $(\text{PocStFoll}[j] \& (\text{MaxPicOrderCntLsb} - 1))$.
- There shall be no j in the range of 0 to $\text{NumPocLtFoll} - 1$, inclusive, where j is not equal to i , for which $\text{PocLtFoll}[i]$ is equal to $(\text{PocLtFoll}[j] \& (\text{MaxPicOrderCntLsb} - 1))$.
- There shall be no j in the range of 0 to $\text{NumPocLtCurr} - 1$, inclusive, for which $\text{PocLtFoll}[i]$ is equal to $(\text{PocLtCurr}[j] \& (\text{MaxPicOrderCntLsb} - 1))$.

The variable NumPicTotalCurr is derived as specified in clause 7.4.7.2. It is a requirement of bitstream conformance that the following applies to the value of NumPicTotalCurr :

- If the current picture is a BLA or CRA picture, the value of NumPicTotalCurr shall be equal to $\text{pps_curr_pic_ref_enabled_flag}$.
- Otherwise, when the current picture contains a P or B slice, the value of NumPicTotalCurr shall not be equal to 0.

The RPS of the current picture consists of five RPS lists; $\text{RefPicSetStCurrBefore}$, $\text{RefPicSetStCurrAfter}$, RefPicSetStFoll , RefPicSetLtCurr and RefPicSetLtFoll . $\text{RefPicSetStCurrBefore}$, $\text{RefPicSetStCurrAfter}$ and RefPicSetStFoll are collectively referred to as the short-term RPS. RefPicSetLtCurr and RefPicSetLtFoll are collectively referred to as the long-term RPS.

NOTE 3 – $\text{RefPicSetStCurrBefore}$, $\text{RefPicSetStCurrAfter}$ and RefPicSetLtCurr contain all reference pictures that may be used for inter prediction of the current picture and one or more pictures that follow the current picture in decoding order. RefPicSetStFoll and RefPicSetLtFoll consist of all reference pictures that are *not* used for inter prediction of the current picture but may be used in inter prediction for one or more pictures that follow the current picture in decoding order.

The derivation process for the RPS and picture marking are performed according to the following ordered steps:

1. The following applies:

```

for( i = 0; i < NumPocLtCurr; i++ )
    if( !CurrDeltaPocMsbPresentFlag[ i ] )
        if( there is a reference picture picX in the DPB with
            PicOrderCntVal & ( MaxPicOrderCntLsb - 1 )
                equal to PocLtCurr[ i ] and nuh_layer_id equal to currPicLayerId )
            RefPicSetLtCurr[ i ] = picX
        else
            RefPicSetLtCurr[ i ] = "no reference picture"
    else
        if( there is a reference picture picX in the DPB with PicOrderCntVal equal to PocLtCurr[ i ]
            and nuh_layer_id equal to currPicLayerId )
            RefPicSetLtCurr[ i ] = picX
        else
            RefPicSetLtCurr[ i ] = "no reference picture"

```

(8-6)

```

for( i = 0; i < NumPocLtFoll; i++ )
    if( !FollDeltaPocMsbPresentFlag[ i ] )
        if( there is a reference picture picX in the DPB with

```



```

PicOrderCntVal & ( MaxPicOrderCntLsb - 1 )
    equal to PocLtFoll[ i ] and nuh_layer_id equal to currPicLayerId )
    RefPicSetLtFoll[ i ] = picX
else
    RefPicSetLtFoll[ i ] = "no reference picture"
else
    if( there is a reference picture picX in the DPB with PicOrderCntVal equal to PocLtFoll[ i ]
        and nuh_layer_id equal to currPicLayerId )
        RefPicSetLtFoll[ i ] = picX
    else
        RefPicSetLtFoll[ i ] = "no reference picture"

```

2. All reference pictures that are included in RefPicSetLtCurr or RefPicSetLtFoll and have nuh_layer_id equal to currPicLayerId are marked as "used for long-term reference".

3. The following applies:

```

for( i = 0; i < NumPocStCurrBefore; i++ )
    if( there is a short-term reference picture picX in the DPB
        with PicOrderCntVal equal to PocStCurrBefore[ i ] and nuh_layer_id equal to
currPicLayerId )
        RefPicSetStCurrBefore[ i ] = picX
    else
        RefPicSetStCurrBefore[ i ] = "no reference picture"

```

```

for( i = 0; i < NumPocStCurrAfter; i++ )
    if( there is a short-term reference picture picX in the DPB
        with PicOrderCntVal equal to PocStCurrAfter[ i ] and nuh_layer_id equal to
currPicLayerId )
        RefPicSetStCurrAfter[ i ] = picX
    else
        RefPicSetStCurrAfter[ i ] = "no reference picture"

```

(8-7)

```

for( i = 0; i < NumPocStFoll; i++ )
    if( there is a short-term reference picture picX in the DPB
        with PicOrderCntVal equal to PocStFoll[ i ] and nuh_layer_id equal to currPicLayerId )
        RefPicSetStFoll[ i ] = picX
    else
        RefPicSetStFoll[ i ] = "no reference picture"

```

4. All reference pictures in the DPB that are not included in RefPicSetLtCurr, RefPicSetLtFoll, RefPicSetStCurrBefore, RefPicSetStCurrAfter, or RefPicSetStFoll and have nuh_layer_id equal to currPicLayerId are marked as "unused for reference".

NOTE 4 – There may be one or more entries in the RPS lists that are equal to "no reference picture" because the corresponding pictures are not present in the DPB. Entries in RefPicSetStFoll or RefPicSetLtFoll that are equal to "no reference picture" should be ignored. An unintentional picture loss should be inferred for each entry in RefPicSetStCurrBefore, RefPicSetStCurrAfter, or RefPicSetLtCurr that is equal to "no reference picture".

NOTE 5 – A picture cannot be included in more than one of the five RPS lists.

It is a requirement of bitstream conformance that the RPS is restricted as follows:

- There shall be no entry in RefPicSetStCurrBefore, RefPicSetStCurrAfter or RefPicSetLtCurr for which one or more of the following are true:
 - The entry is equal to "no reference picture".
 - The entry is an SLNR picture and has TemporalId equal to that of the current picture.
 - The entry is a picture that has TemporalId greater than that of the current picture.
- There shall be no entry in RefPicSetLtCurr or RefPicSetLtFoll for which the difference between the picture order count value of the current picture and the picture order count value of the entry is greater than or equal to 2^{24} .

- When the current picture is a temporal sub-layer access (TSA) picture, there shall be no picture included in the RPS with TemporalId greater than or equal to the TemporalId of the current picture.
- When the current picture is a step-wise temporal sub-layer access (STSA) picture, there shall be no picture included in RefPicSetStCurrBefore, RefPicSetStCurrAfter or RefPicSetLtCurr that has TemporalId equal to that of the current picture.
- When the current picture is a picture that follows, in decoding order, an STSA picture that has TemporalId equal to that of the current picture, there shall be no picture that has TemporalId equal to that of the current picture included in RefPicSetStCurrBefore, RefPicSetStCurrAfter or RefPicSetLtCurr that precedes the STSA picture in decoding order.
- When the current picture is a CRA picture, there shall be no picture included in the RPS that precedes, in output order or decoding order, any preceding IRAP picture in decoding order (when present).
- When the current picture is a trailing picture, there shall be no picture in RefPicSetStCurrBefore, RefPicSetStCurrAfter or RefPicSetLtCurr that was generated by the decoding process for generating unavailable reference pictures as specified in clause 8.3.3.
- When the current picture is a trailing picture, there shall be no picture in the RPS that precedes the associated IRAP picture in output order or decoding order.
- When the current picture is a RADL picture, there shall be no picture included in RefPicSetStCurrBefore, RefPicSetStCurrAfter or RefPicSetLtCurr that is any of the following:
 - A RASL picture
 - A picture that was generated by the decoding process for generating unavailable reference pictures as specified in clause 8.3.3
 - A picture that precedes the associated IRAP picture in decoding order
- When sps_temporal_id_nesting_flag is equal to 1, the following applies:
 - Let tIdA be the value of TemporalId of the current picture picA.
 - Any picture picB with TemporalId equal to tIdB that is less than or equal to tIdA shall not be included in RefPicSetStCurrBefore, RefPicSetStCurrAfter or RefPicSetLtCurr of picA when there exists a picture picC that has TemporalId less than tIdB, follows picB in decoding order, and precedes picA in decoding order.

8.3.3 Decoding process for generating unavailable reference pictures

8.3.3.1 General decoding process for generating unavailable reference pictures

This process is invoked once per coded picture when the current picture is a BLA picture or is a CRA picture with NoRasOutputFlag equal to 1.

NOTE – This process is primarily specified only for the specification of syntax constraints for RASL pictures. The entire specification of the decoding process for RASL pictures associated with an IRAP picture that has NoRasOutputFlag equal to 1 is included herein only for purposes of specifying constraints on the allowed syntax content of such RASL pictures. During the decoding process, any RASL pictures associated with an IRAP picture that has NoRasOutputFlag equal to 1 may be ignored, as these pictures are not specified for output and have no effect on the decoding process of any other pictures that are specified for output. However, in HRD operations as specified in Annex C, RASL access units may need to be taken into consideration in derivation of coded picture buffer (CPB) arrival and removal times.

When this process is invoked, the following applies:

- For each RefPicSetStFoll[i], with i in the range of 0 to NumPocStFoll – 1, inclusive, that is equal to "no reference picture", a picture is generated as specified in clause 8.3.3.2, and the following applies:
 - The value of PicOrderCntVal for the generated picture is set equal to PocStFoll[i].
 - The value of PicOutputFlag for the generated picture is set equal to 0.
 - The generated picture is marked as "used for short-term reference".
 - RefPicSetStFoll[i] is set to be the generated reference picture.
 - The value of nuh_layer_id for the generated picture is set equal to nuh_layer_id of the current picture.
- For each RefPicSetLtFoll[i], with i in the range of 0 to NumPocLtFoll – 1, inclusive, that is equal to "no reference picture", a picture is generated as specified in clause 8.3.3.2, and the following applies:
 - The value of PicOrderCntVal for the generated picture is set equal to PocLtFoll[i].

- The value of slice_pic_order_cnt_lsb for the generated picture is inferred to be equal to (PocLtFoll[i] & (MaxPicOrderCntLsb – 1)).
- The value of PicOutputFlag for the generated picture is set equal to 0.
- The generated picture is marked as "used for long-term reference".
- RefPicSetLtFoll[i] is set to be the generated reference picture.
- The value of nuh_layer_id for the generated picture is set equal to nuh_layer_id of the current picture.

8.3.3.2 Generation of one unavailable picture

When this process is invoked, an unavailable picture is generated as follows:

- The value of each element in the sample array S_L for the picture is set equal to $1 \ll (\text{BitDepth}_Y - 1)$.
- When ChromaArrayType is not equal to 0, the value of each element in the sample arrays S_{Cb} and S_{Cr} for the picture is set equal to $1 \ll (\text{BitDepth}_C - 1)$.
- The prediction mode CuPredMode[x][y] is set equal to MODE_INTRA for $x = 0..\text{pic_width_in_luma_samples} - 1$, $y = 0..\text{pic_height_in_luma_samples} - 1$.

8.3.4 Decoding process for reference picture lists construction

This process is invoked at the beginning of the decoding process for each P or B slice.

Reference pictures are addressed through reference indices as specified in clause 8.5.3.3.2. A reference index is an index into a reference picture list. When decoding a P slice, there is a single reference picture list RefPicList0. When decoding a B slice, there is a second independent reference picture list RefPicList1 in addition to RefPicList0.

At the beginning of the decoding process for each slice, the reference picture lists RefPicList0 and, for B slices, RefPicList1 are derived.

The variable NumRpsCurrTempList0 is set equal to $\text{Max}(\text{num_ref_idx_l0_active_minus1} + 1, \text{NumPicTotalCurr})$ and the list RefPicListTemp0 is constructed as follows:

If TwoVersionsOfCurrDecPicFlag is equal to 1, let the variable currPic be the current decoded picture before the invocation of the in-loop filter process; otherwise (TwoVersionsOfCurrDecPicFlag is equal to 0), let the variable currPic be the current decoded picture after the invocation of the in-loop filter process. The variable NumRpsCurrTempList0 is set equal to $\text{Max}(\text{num_ref_idx_l0_active_minus1} + 1, \text{NumPicTotalCurr})$ and the list RefPicListTemp0 is constructed as follows:

```

rIdx = 0
while( rIdx < NumRpsCurrTempList0 ) {
    for( i = 0; i < NumPocStCurrBefore && rIdx < NumRpsCurrTempList0; rIdx++, i++ )
        RefPicListTemp0[ rIdx ] = RefPicSetStCurrBefore[ i ]
    for( i = 0; i < NumPocStCurrAfter && rIdx < NumRpsCurrTempList0; rIdx++, i++ )           (8-8)
        RefPicListTemp0[ rIdx ] = RefPicSetStCurrAfter[ i ]
    for( i = 0; i < NumPocLtCurr && rIdx < NumRpsCurrTempList0; rIdx++, i++ )
        RefPicListTemp0[ rIdx ] = RefPicSetLtCurr[ i ]
    if( pps_curr_pic_ref_enabled_flag )
        RefPicListTemp0[ rIdx++ ] = currPic
}

```

The list RefPicList0 is constructed as follows:

```

for( rIdx = 0; rIdx <= num_ref_idx_l0_active_minus1; rIdx++ )
    RefPicList0[ rIdx ] = ref_pic_list_modification_flag_l0 ? RefPicListTemp0[ list_entry_l0[ rIdx ] ] :
                                                                RefPicListTemp0[ rIdx ]
if( pps_curr_pic_ref_enabled_flag && !ref_pic_list_modification_flag_l0 &&           (8-9)
    NumRpsCurrTempList0 > ( num_ref_idx_l0_active_minus1 + 1 ) )
    RefPicList0[ num_ref_idx_l0_active_minus1 ] = currPic

```

When the slice is a B slice, the variable NumRpsCurrTempList1 is set equal to $\text{Max}(\text{num_ref_idx_l1_active_minus1} + 1, \text{NumPicTotalCurr})$ and the list RefPicListTemp1 is constructed as follows:

```

rIdx = 0
while( rIdx < NumRpsCurrTempList1 ) {
    for( i = 0; i < NumPocStCurrAfter && rIdx < NumRpsCurrTempList1; rIdx++, i++ )
        RefPicListTemp1[ rIdx ] = RefPicSetStCurrAfter[ i ]
    for( i = 0; i < NumPocStCurrBefore && rIdx < NumRpsCurrTempList1; rIdx++, i++ )      (8-10)
        RefPicListTemp1[ rIdx ] = RefPicSetStCurrBefore[ i ]
    for( i = 0; i < NumPocLtCurr && rIdx < NumRpsCurrTempList1; rIdx++, i++ )
        RefPicListTemp1[ rIdx ] = RefPicSetLtCurr[ i ]
    if( pps_curr_pic_ref_enabled_flag )
        RefPicListTemp1[ rIdx++ ] = currPic
}

```

When the slice is a B slice, the list RefPicList1 is constructed as follows:

```

for( rIdx = 0; rIdx <= num_ref_idx_l1_active_minus1; rIdx++ )
    RefPicList1[ rIdx ] = ref_pic_list_modification_flag_l1 ? RefPicListTemp1[ list_entry_l1[ rIdx ] ] :
                                                                RefPicListTemp1[ rIdx ]
(8-11)

```

It is a requirement of bitstream conformance that when nuh_layer_id is equal to 0, nal_unit_type has a value in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive (i.e. the picture is an IRAP picture), pps_curr_pic_ref_enabled_flag is equal to 1, and slice_type is not equal to 2, RefPicList0 and RefPicList1 shall not contain entries that refer to a picture other than the current picture.

8.3.5 Decoding process for collocated picture and no backward prediction flag

This process is invoked at the beginning of the decoding process for each P or B slice, after decoding of the slice header as well as the invocation of the decoding process for reference picture set as specified in clause 8.3.2 and the invocation of the decoding process for reference picture list construction for the slice as specified in clause 8.3.4, but prior to the decoding of any coding unit.

When slice_temporal_mvp_enabled_flag is equal to 1, the variable ColPic is derived as follows:

- If slice_type is equal to B and collocated_from_l0_flag is equal to 0, ColPic is set equal to RefPicList1[collocated_ref_idx].
- Otherwise (slice_type is equal to B and collocated_from_l0_flag is equal to 1, or slice_type is equal to P), ColPic is set equal to RefPicList0[collocated_ref_idx].

The variable NoBackwardPredFlag is derived as follows:

- If DiffPicOrderCnt(aPic, CurrPic) is less than or equal to 0 for each picture aPic in RefPicList0 or RefPicList1 of the current slice, NoBackwardPredFlag is set equal to 1.
- Otherwise, NoBackwardPredFlag is set equal to 0.

8.4 Decoding process for coding units coded in intra prediction mode

8.4.1 General decoding process for coding units coded in intra prediction mode

Inputs to this process are:

- a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a variable log2CbSize specifying the size of the current luma coding block.

Output of this process is a modified reconstructed picture before deblocking filtering.

The derivation process for quantization parameters as specified in clause 8.6.1 is invoked with the luma location (xCb, yCb) as input.

A variable nCbS is set equal to $1 \ll \log_2 \text{CbSize}$.

When residual_adaptive_colour_transform_enabled_flag is equal to 1, the residual sample arrays resSamplesL, resSamplesCb, and resSamplesCr store the residual samples of the current coding unit.

Depending on the values of `pcm_flag[xCb][yCb]`, `palette_mode_flag[xCb][yCb]`, and `IntraSplitFlag`, the decoding process for luma samples is specified as follows:

- If `pcm_flag[xCb][yCb]` is equal to 1, the reconstructed picture is modified as follows:

$$S_L[xCb + i][yCb + j] = \text{pcm_sample_luma}[(nCbS * j) + i] \ll (\text{BitDepth}_Y - \text{PcmBitDepth}_Y), \text{ with } i, j = 0..nCbS - 1 \quad (8-12)$$

- Otherwise (`pcm_flag[xCb][yCb]` is equal to 0), if `palette_mode_flag[xCb][yCb]` is equal to 1, the following ordered steps apply:

1. The decoding process for palette intra blocks as specified in clause 8.4.4.2.7 is invoked with the luma location (`xCb`, `yCb`), the variable `cIdx` set equal to 0, and `nCbSX` and `nCbSY` both set equal to `nCbS` as inputs, and the output is an `nCbS` x `nCbS` array of reconstructed palette sample values, `recSamples[x][y]`, `x`, `y` = 0..`nCbS` – 1.
2. The reconstructed picture is modified as follows for `x`, `y` = 0..`nCbS` – 1 and `y` = 0..`nCbS` – 1:

- If `palette_transpose_flag` is equal to 1, the following applies:

$$S_L[xCb + x][yCb + y] = \text{recSamples}[y][x] \quad (8-13)$$

- Otherwise (`palette_transpose_flag` is equal to 0), the following applies:

$$S_L[xCb + x][yCb + y] = \text{recSamples}[x][y] \quad (8-14)$$

- Otherwise (`pcm_flag[xCb][yCb]` is equal to 0, `palette_mode_flag[xCb][yCb]` is equal to 0), if `IntraSplitFlag` is equal to 0, the following ordered steps apply:

1. The derivation process for the intra prediction mode as specified in clause 8.4.2 is invoked with the luma location (`xCb`, `yCb`) as input.
2. If `residual_adaptive_colour_transform_enabled_flag` is equal to 1, the following applies:
 - The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the luma location (`xCb`, `yCb`), the variable `log2TrafoSize` set equal to `log2CbSize`, the variable `trafoDepth` set equal to 0, the variable `predModeIntra` set equal to `IntraPredModeY[xCb][yCb]`, the variable `cIdx` set equal to 0 and variable `controlParaAct` set equal to 1 as inputs, and the output is a modified residual sample array `resSamplesL`.
 - The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the chroma location (`xCb`, `yCb`), the variable `log2TrafoSize` set equal to `log2CbSize`, the variable `trafoDepth` set equal to 0, the variable `predModeIntra` set equal to `IntraPredModeC`, the variable `cIdx` set equal to 1 and variable `controlParaAct` set equal to 1 as inputs, and the output is a modified residual sample array `resSamplesCb`.
 - The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the chroma location (`xCb`, `yCb`), the variable `log2TrafoSize` set equal to `log2CbSize`, the variable `trafoDepth` set equal to 0, the variable `predModeIntra` set equal to `IntraPredModeC`, the variable `cIdx` set equal to 2 and variable `controlParaAct` set equal to 1 as inputs, and the output is a modified residual sample array `resSamplesCr`.
 - The residual modification process for blocks using adaptive colour transform as specified in clause 8.6.8 is invoked with location (`xCb`, `yCb`), the variable `log2TrafoSize` set equal to `log2CbSize`, the variable `trafoDepth` set equal to 0, the variable `resSampleArrayL` set equal to `resSamplesL`, the variable `resSampleArrayCb` set equal to `resSamplesCb` and the variable `resSampleArrayCr` set equal to `resSamplesCr` as inputs, and the outputs are modified versions of `resSampleL`, `resSampleCb` and `resSampleCr`.
3. The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the luma location (`xCb`, `yCb`), the variable `log2TrafoSize` set equal to `log2CbSize`, the variable `trafoDepth` set equal to 0, the variable `predModeIntra` set equal to `IntraPredModeY[xCb][yCb]`, the variable `cIdx` set equal to 0, the variable `controlParaAct` set equal to (`residual_adaptive_colour_transform_enabled_flag ? 2 : 0`), and the array `resSamplesL` when `controlParaAct` is equal to 2 as inputs, and the output is a modified reconstructed picture before deblocking filtering.
- Otherwise (`pcm_flag[xCb][yCb]` is equal to 0, `palette_mode_flag[xCb][yCb]` is equal to 0, and `IntraSplitFlag` is equal to 1), for the variable `blkIdx` proceeding over the values 0..3, the following ordered steps apply:
 1. The variable `xPb` is set equal to `xCb + (nCbS >> 1) * (blkIdx % 2)`.

2. The variable y_{Pb} is set equal to $y_{Cb} + (n_{CbS} \gg 1) * (blkIdx / 2)$.
3. The derivation process for the intra prediction mode as specified in clause 8.4.2 is invoked with the luma location (x_{Pb}, y_{Pb}) as input.
4. If `residual_adaptive_colour_transform_enabled_flag` is equal to 1, the following applies:
 - The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the luma location (x_{Pb}, y_{Pb}) , the variable $\log_2TrafoSize$ set equal to $\log_2CbSize - 1$, the variable $trafoDepth$ set equal to 1, the variable $predModeIntra$ set equal to $IntraPredModeY[x_{Pb}][y_{Pb}]$, the variable $cIdx$ set equal to 0 and variable $controlParaAct$ equal to 1 as inputs, and the output is a modified residual sample array $resSamplesL$.
 - The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the chroma location (x_{Pb}, y_{Pb}) , the variable $\log_2TrafoSize$ set equal to $\log_2CbSize - 1$, the variable $trafoDepth$ set equal to 1, the variable $predModeIntra$ set equal to $IntraPredModeC$, the variable $cIdx$ set equal to 1 and variable $controlParaAct$ equal to 1 as inputs, and the output is a modified residual sample array $resSamplesCb$.
 - The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the chroma location (x_{Pb}, y_{Pb}) , the variable $\log_2TrafoSize$ set equal to $\log_2CbSize - 1$, the variable $trafoDepth$ set equal to 1, the variable $predModeIntra$ set equal to $IntraPredModeC$, the variable $cIdx$ set equal to 2 and variable $controlParaAct$ set equal to 1 as inputs, and the output is a modified residual sample array $resSamplesCr$.
 - The residual modification process for blocks using adaptive colour transform as specified in clause 8.6.8 is invoked with location (x_{Cb}, y_{Cb}) , the variable $\log_2TrafoSize$ set equal to $\log_2CbSize - 1$, the variable $trafoDepth$ set equal to 1, the variable $resSampleArrayL$ set equal to $resSamplesL$, the variable $resSampleArrayCb$ set equal to $resSamplesCb$ and the variable $resSampleArrayCr$ set equal to $resSamplesCr$ as inputs, and the outputs are modified versions of $resSampleL$, $resSampleCb$ and $resSampleCr$.
5. The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the luma location (x_{Pb}, y_{Pb}) , the variable $\log_2TrafoSize$ set equal to $\log_2CbSize - 1$, the variable $trafoDepth$ set equal to 1, the variable $predModeIntra$ set equal to $IntraPredModeY[x_{Pb}][y_{Pb}]$, the variable $cIdx$ set equal to 0, the variable $controlParaAct$ set equal to $(residual_adaptive_colour_transform_enabled_flag ? 2 : 0)$, and the array $resSamplesL$ when $controlParaAct$ is equal to 2 as inputs, and the output is a modified reconstructed picture before deblocking filtering.

When `ChromaArrayType` is not equal to 0, the following applies.

The variable $\log_2CbSizeC$ is set equal to $\log_2CbSize - (ChromaArrayType == 3 ? 0 : 1)$.

Depending on the values of `pcm_flag[xCb][yCb]` and `IntraSplitFlag`, the decoding process for chroma samples is specified as follows:

- If `pcm_flag[xCb][yCb]` is equal to 1, the reconstructed picture is modified as follows:

$$S_{Cb}[xCb / SubWidthC + i][yCb / SubHeightC + j] = \text{pcm_sample_chroma}[(n_{CbS} / SubWidthC * j) + i] \ll (BitDepth_C - PcmBitDepth_C),$$

with $i = 0..n_{CbS} / SubWidthC - 1$ and $j = 0..n_{CbS} / SubHeightC - 1$ (8-15)

$$S_{Cr}[xCb / SubWidthC + i][yCb / SubHeightC + j] = \text{pcm_sample_chroma}[(n_{CbS} / SubWidthC * (j + n_{CbS} / SubHeightC)) + i] \ll (BitDepth_C - PcmBitDepth_C),$$

with $i = 0..n_{CbS} / SubWidthC - 1$ and $j = 0..n_{CbS} / SubHeightC - 1$ (8-16)

- Otherwise (`pcm_flag[xCb][yCb]` is equal to 0), if `palette_mode_flag[xCb][yCb]` is equal to 1 the following ordered steps apply:

1. n_{CbSX} is set as follows:

- If `palette_transpose_flag` is equal to 1, n_{CbSX} is set equal to $(n_{CbS} / SubHeightC)$, n_{CbY} is set equal to $(n_{CbS} / SubWidthC)$.
- Otherwise (`palette_transpose_flag` is equal to 0) n_{CbSX} is set equal to $(n_{CbS} / SubWidthC)$, n_{CbY} is set equal to $(n_{CbS} / SubHeightC)$.

2. The decoding process for palette intra blocks as specified in clause 8.4.4.2.7 is invoked with the chroma location (x_{Cb}, y_{Cb}) , the variable $cIdx$ set equal to 1, n_{CbSX} , and n_{CbSY} as inputs, and the output is an $(n_{CbS} / SubWidthC) \times (n_{CbS} / SubHeightC)$ array of reconstructed palette sample values, $recSamples[x][y]$, $x = 0 \dots n_{CbS} / SubWidthC - 1$, $y = 0..n_{CbS} / SubHeightC - 1$, when `palette_transpose_flag` is equal to 0, or an

$(nCbS / SubHeightC) \times (nCbS / SubWidthC)$ array of reconstructed palette sample values, $recSamples[x][y]$, $x = 0 \dots nCbS / SubWidthC - 1$, $y = 0 \dots nCbS / SubHeightC - 1$ when `palette_transpose_flag` is equal to 1.

3. The reconstructed picture is modified as follows:

- If `palette_transpose_flag` is equal to 1, the following applies:

$$SCb[xCb / SubWidthC + x][yCb / SubHeightC + y] = recSamples[y][x],$$

with $x = 0 \dots nCbS / SubWidthC - 1$ and $y = 0 \dots nCbS / SubHeightC - 1$ (8-17)

- Otherwise (`palette_transpose_flag` is equal to 0), the following applies:

$$SCb[xCb / SubWidthC + x][yCb / SubHeightC + y] = recSamples[x][y],$$

with $x = 0 \dots nCbS / SubWidthC - 1$ and $y = 0 \dots nCbS / SubHeightC - 1$ (8-18)

4. The decoding process for palette intra blocks as specified in clause 8.4.4.2.7 is invoked with the chroma location (xCb, yCb) , the variable `cIdx` set equal to 2, `nCbSX` set equal to $nCbS / SubWidthC$, and `nCbSY` set equal to $nCbS / SubHeightC$ as inputs, and the output is an $(nCbS / SubWidthC) \times (nCbS / SubHeightC)$ array of reconstructed palette sample values, $recSamples[x][y]$, $x = 0 \dots nCbS / SubWidthC - 1$, $y = 0 \dots nCbS / SubHeightC - 1$, when `palette_transpose_flag` is equal to 0, or an $(nCbS / SubHeightC) \times (nCbS / SubWidthC)$ array of reconstructed palette sample values, $recSamples[x][y]$, $x = 0 \dots nCbS / SubHeightC - 1$, $y = 0 \dots nCbS / SubWidthC - 1$, when `palette_transpose_flag` is equal to 1.

5. The reconstructed picture is modified as follows:

- If `palette_transpose_flag` is equal to 1, the following applies:

$$SCr[xCb / SubWidthC + x][yCb / SubHeightC + y] = recSamples[y][x],$$

with $x = 0 \dots nCbS / SubWidthC - 1$ and $y = 0 \dots nCbS / SubHeightC - 1$ (8-19)

- Otherwise (`palette_transpose_flag` is equal to 0), the following applies:

$$SCr[xCb / SubWidthC + x][yCb / SubHeightC + y] = recSamples[x][y],$$

with $x = 0 \dots nCbS / SubWidthC - 1$ and $y = 0 \dots nCbS / SubHeightC - 1$ (8-20)

- Otherwise (`pcm_flag[xCb][yCb]` is equal to 0 and `palette_mode_flag[xCb][yCb]` is equal to 0), if `IntraSplitFlag` is equal to 0 or `ChromaArrayType` is not equal to 3, the following ordered steps apply:

1. The derivation process for the chroma intra prediction mode as specified in clause 8.4.3 is invoked with the luma location (xCb, yCb) as input, and the output is the variable `IntraPredModeC`.
2. The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the chroma location $(xCb / SubWidthC, yCb / SubHeightC)$, the variable `log2TrafoSize` set equal to `log2CbSizeC`, the variable `trafoDepth` set equal to 0, the variable `predModeIntra` set equal to `IntraPredModeC`, the variable `cIdx` set equal to 1, the variable `controlParaAct` set equal to $(residual_adaptive_colour_transform_enabled_flag ? 2 : 0)$, and the array `resSamplesCb` when `controlParaAct` is equal to 2 as inputs, and the output is a modified reconstructed picture before deblocking filtering.
3. The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the chroma location $(xCb / SubWidthC, yCb / SubHeightC)$, the variable `log2TrafoSize` set equal to `log2CbSizeC`, the variable `trafoDepth` set equal to 0, the variable `predModeIntra` set equal to `IntraPredModeC`, the variable `cIdx` set equal to 2, the variable `controlParaAct` set equal to $(residual_adaptive_colour_transform_enabled_flag ? 2 : 0)$, and the array `resSamplesCr` when `controlParaAct` is equal to 2 as inputs, and the output is a modified reconstructed picture before deblocking filtering.

- Otherwise (`pcm_flag[xCb][yCb]` is equal to 0, `palette_mode_flag[xCb][yCb]` is equal to 0, `IntraSplitFlag` is equal to 1 and `ChromaArrayType` is equal to 3), for the variable `blkIdx` proceeding over the values 0..3, the following ordered steps apply:

1. The variable `xPb` is set equal to $xCb + (nCbS \gg 1) * (blkIdx \% 2)$.
2. The variable `yPb` is set equal to $yCb + (nCbS \gg 1) * (blkIdx / 2)$.
3. The derivation process for the chroma intra prediction mode as specified in clause 8.4.3 is invoked with the luma location (xPb, yPb) as input, and the output is the variable `IntraPredModeC`.

4. The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the chroma location (xPb, yPb), the variable log2TrafoSize set equal to log2CbSizeC – 1, the variable trafoDepth set equal to 1, the variable predModeIntra set equal to IntraPredModeC, the variable cIdx set equal to 1, the variable controlParaAct set equal to (residual_adaptive_colour_transform_enabled_flag ? 2 : 0), and the array resSamplesCb when controlParaAct is equal to 2 as inputs, and the output is a modified reconstructed picture before deblocking filtering.
5. The general decoding process for intra blocks as specified in clause 8.4.4.1 is invoked with the chroma location (xPb, yPb), the variable log2TrafoSize set equal to log2CbSizeC – 1, the variable trafoDepth set equal to 1, the variable predModeIntra set equal to IntraPredModeC, the variable cIdx set equal to 2, the variable controlParaAct set equal to (residual_adaptive_colour_transform_enabled_flag ? 2 : 0), and the array resSamplesCr when controlParaAct is equal to 2 as inputs, and the output is a modified reconstructed picture before deblocking filtering.

8.4.2 Derivation process for luma intra prediction mode

Input to this process is a luma location (xPb, yPb) specifying the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture.

In this process, the luma intra prediction mode IntraPredModeY[xPb][yPb] is derived.

Table 8-1 specifies the value for the intra prediction mode and the associated names.

Table 8-1 – Specification of intra prediction mode and associated names

Intra prediction mode	Associated name
0	INTRA_PLANAR
1	INTRA_DC
2..34	INTRA_ANGULAR2..INTRA_ANGULAR34

IntraPredModeY[xPb][yPb] labelled 0..34 represents directions of predictions as illustrated in Figure 8-1.

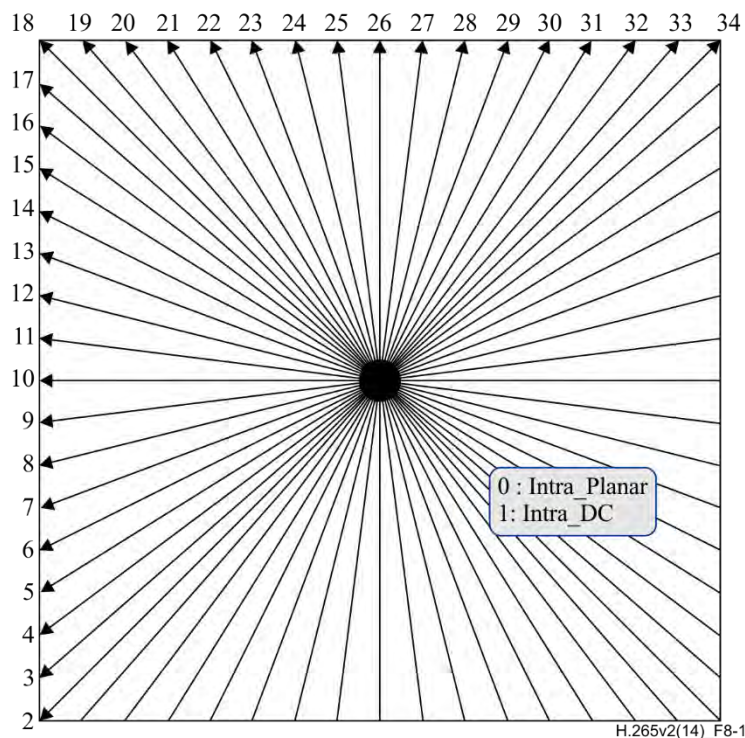


Figure 8-1 – Intra prediction mode directions (informative)

IntraPredModeY[xPb][yPb] is derived by the following ordered steps:

1. The neighbouring locations (xNbA, yNbA) and (xNbB, yNbB) are set equal to (xPb – 1, yPb) and (xPb, yPb – 1), respectively.
2. For X being replaced by either A or B, the variables candIntraPredModeX are derived as follows:
 - The availability derivation process for a block in z-scan order as specified in clause 6.4.1 is invoked with the location (xCurr, yCurr) set equal to (xPb, yPb) and the neighbouring location (xNbY, yNbY) set equal to (xNbX, yNbX) as inputs, and the output is assigned to availableX.
 - The candidate intra prediction mode candIntraPredModeX is derived as follows:
 - If availableX is equal to FALSE, candIntraPredModeX is set equal to INTRA_DC.
 - Otherwise, if CuPredMode[xNbX][yNbX] is not equal to MODE_INTRA or pcm_flag[xNbX][yNbX] is equal to 1, candIntraPredModeX is set equal to INTRA_DC,
 - Otherwise, if X is equal to B and yPb – 1 is less than ((yPb >> CtbLog2SizeY) << CtbLog2SizeY), candIntraPredModeB is set equal to INTRA_DC.
 - Otherwise, candIntraPredModeX is set equal to IntraPredModeY[xNbX][yNbX].
3. The candModeList[x] with x = 0..2 is derived as follows:
 - If candIntraPredModeB is equal to candIntraPredModeA, the following applies:
 - If candIntraPredModeA is less than 2 (i.e., equal to INTRA_PLANAR or INTRA_DC), candModeList[x] with x = 0..2 is derived as follows:

$$\text{candModeList}[0] = \text{INTRA_PLANAR} \quad (8-21)$$

$$\text{candModeList}[1] = \text{INTRA_DC} \quad (8-22)$$

$$\text{candModeList}[2] = \text{INTRA_ANGULAR26} \quad (8-23)$$
 - Otherwise, candModeList[x] with x = 0..2 is derived as follows:

$$\text{candModeList}[0] = \text{candIntraPredModeA} \quad (8-24)$$

$$\text{candModeList}[1] = 2 + ((\text{candIntraPredModeA} + 29) \% 32) \quad (8-25)$$

$$\text{candModeList}[2] = 2 + ((\text{candIntraPredModeA} - 2 + 1) \% 32) \quad (8-26)$$
 - Otherwise (candIntraPredModeB is not equal to candIntraPredModeA), the following applies:
 - candModeList[0] and candModeList[1] are derived as follows:

$$\text{candModeList}[0] = \text{candIntraPredModeA} \quad (8-27)$$

$$\text{candModeList}[1] = \text{candIntraPredModeB} \quad (8-28)$$
 - If neither of candModeList[0] and candModeList[1] is equal to INTRA_PLANAR, candModeList[2] is set equal to INTRA_PLANAR,
 - Otherwise, if neither of candModeList[0] and candModeList[1] is equal to INTRA_DC, candModeList[2] is set equal to INTRA_DC,
 - Otherwise, candModeList[2] is set equal to INTRA_ANGULAR26.
4. IntraPredModeY[xPb][yPb] is derived by applying the following procedure:
 - If prev_intra_luma_pred_flag[xPb][yPb] is equal to 1, the IntraPredModeY[xPb][yPb] is set equal to candModeList[mpm_idx[xPb][yPb]].
 - Otherwise, IntraPredModeY[xPb][yPb] is derived by applying the following ordered steps:
 - 1) The array candModeList[x], x = 0..2 is modified as the following ordered steps:
 - i. When candModeList[0] is greater than candModeList[1], both values are swapped as follows:

$$(\text{candModeList}[0], \text{candModeList}[1]) = \text{Swap}(\text{candModeList}[0], \text{candModeList}[1]) \quad (8-29)$$

- ii. When $\text{candModeList}[0]$ is greater than $\text{candModeList}[2]$, both values are swapped as follows:

$$(\text{candModeList}[0], \text{candModeList}[2]) = \text{Swap}(\text{candModeList}[0], \text{candModeList}[2]) \quad (8-30)$$

- iii. When $\text{candModeList}[1]$ is greater than $\text{candModeList}[2]$, both values are swapped as follows:

$$(\text{candModeList}[1], \text{candModeList}[2]) = \text{Swap}(\text{candModeList}[1], \text{candModeList}[2]) \quad (8-31)$$

2) $\text{IntraPredModeY}[\text{xPb}][\text{yPb}]$ is derived by the following ordered steps:

- $\text{IntraPredModeY}[\text{xPb}][\text{yPb}]$ is set equal to $\text{rem_intra_luma_pred_mode}[\text{xPb}][\text{yPb}]$.
- For i equal to 0 to 2, inclusive, when $\text{IntraPredModeY}[\text{xPb}][\text{yPb}]$ is greater than or equal to $\text{candModeList}[i]$, the value of $\text{IntraPredModeY}[\text{xPb}][\text{yPb}]$ is incremented by one.

8.4.3 Derivation process for chroma intra prediction mode

This process is only invoked when ChromaArrayType is not equal to 0.

Input to this process is a luma location (xPb , yPb) specifying the top-left sample of the current chroma prediction block relative to the top-left luma sample of the current picture.

Output of this process is the variable IntraPredModeC .

The variable modeIdx is derived using $\text{intra_chroma_pred_mode}[\text{xPb}][\text{yPb}]$ and $\text{IntraPredModeY}[\text{xPb}][\text{yPb}]$ as specified in Table 8-2.

The chroma intra prediction mode IntraPredModeC is derived as follows:

- If ChromaArrayType is equal to 2, IntraPredModeC is set using modeIdx as specified in Table 8-3.
- Otherwise, IntraPredModeC is set equal to modeIdx .

Table 8-2 – Specification of modeIdx

$\text{intra_chroma_pred_mode}[\text{xPb}][\text{yPb}]$	$\text{IntraPredModeY}[\text{xPb}][\text{yPb}]$				
	0	26	10	1	$X (0 \leq X \leq 34)$
0	34	0	0	0	0
1	26	34	26	26	26
2	10	10	34	10	10
3	1	1	1	34	1
4	0	26	10	1	X

Table 8-3 – Specification of IntraPredModeC when ChromaArrayType is equal to 2

modeIdx	X <= 2		3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
IntraPredModeC	X		2	2	2	3	5	7	8	10	12	13	15	17	18	19	20
modeIdx	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
IntraPredModeC	21	22	23	23	24	24	25	25	26	27	27	28	28	29	29	30	31

8.4.4 Decoding process for intra blocks

8.4.4.1 General decoding process for intra blocks

Inputs to this process are:

- a sample location ($xTb0, yTb0$) specifying the top-left sample of the current transform block relative to the top-left sample of the current picture,
- a variable $\log2TrafoSize$ specifying the size of the current transform block,
- a variable $trafoDepth$ specifying the hierarchy depth of the current block relative to the coding unit,
- a variable $predModeIntra$ specifying the intra prediction mode,
- a variable $cIdx$ specifying the colour component of the current block,
- a variable $controlParaAct$ specifying the applicable processes,
- when $controlParaAct$ is equal to 2, a residual sample array $resSamplesRec$ specifying the reconstructed residual samples for the current colour component of the current coding block.

Output of this process is a modified reconstructed picture before deblocking filtering when $controlParaAct$ is not equal to 1, or a modified residual sample array $resSampleArray$ for the current colour component of the current coding block when $controlParaAct$ is equal to 1.

The luma sample location ($xTbY, yTbY$) specifying the top-left sample of the current luma transform block relative to the top-left luma sample of the current picture is derived as follows:

$$(xTbY, yTbY) = (cIdx == 0) ? (xTb0, yTb0) : (xTb0 * SubWidthC, yTb0 * SubHeightC) \quad (8-32)$$

The variable $splitFlag$ is derived as follows:

- If $cIdx$ is equal to 0, $splitFlag$ is set equal to $split_transform_flag[xTbY][yTbY][trafoDepth]$.
- Otherwise, if all of the following conditions are true, $splitFlag$ is set equal to 1.
 - $cIdx$ is greater than 0
 - $split_transform_flag[xTbY][yTbY][trafoDepth]$ is equal to 1
 - $\log2TrafoSize$ is greater than 2
- Otherwise, $splitFlag$ is set equal to 0.

Depending on the value of $splitFlag$, the following applies:

- If $splitFlag$ is equal to 1, the following ordered steps apply:
 1. The variables $xTb1$ and $yTb1$ are derived as follows:
 - If $cIdx$ is equal to 0 or $ChromaArrayType$ is not equal to 2, the following applies:
 - The variable $xTb1$ is set equal to $xTb0 + (1 \ll (\log2TrafoSize - 1))$.
 - The variable $yTb1$ is set equal to $yTb0 + (1 \ll (\log2TrafoSize - 1))$.
 - Otherwise ($ChromaArrayType$ is equal to 2 and $cIdx$ is greater than 0), the following applies:
 - The variable $xTb1$ is set equal to $xTb0 + (1 \ll (\log2TrafoSize - 1))$.
 - The variable $yTb1$ is set equal to $yTb0 + (2 \ll (\log2TrafoSize - 1))$.
 2. The general decoding process for intra blocks as specified in this clause is invoked with the location ($xTb0, yTb0$), the variable $\log2TrafoSize$ set equal to $\log2TrafoSize - 1$, the variable $trafoDepth$ set equal to $trafoDepth + 1$, the intra prediction mode $predModeIntra$, the variable $cIdx$, the variable $controlParaAct$, and the array $resSamplesRec$ when $controlParaAct$ is equal to 2 as inputs and the output is a modified reconstructed picture before deblocking filtering when $controlParaAct$ is not equal to 1 or a modified residual sample array $resSampleArray$ when $controlParaAct$ is equal to 1.
 3. The general decoding process for intra blocks as specified in this clause is invoked with the location ($xTb1, yTb0$), the variable $\log2TrafoSize$ set equal to $\log2TrafoSize - 1$, the variable $trafoDepth$ set equal to $trafoDepth + 1$, the intra prediction mode $predModeIntra$, the variable $cIdx$, the variable $controlParaAct$, and the array $resSamplesRec$ when $controlParaAct$ is equal to 2 as inputs, and the output is a modified reconstructed

picture before deblocking filtering when controlParaAct is not equal to 1 or a modified residual sample array resSampleArray when controlParaAct is equal to 1.

4. The general decoding process for intra blocks as specified in this clause is invoked with the location (xTb0, yTb1), the variable log2TrafoSize set equal to log2TrafoSize – 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode predModeIntra, the variable cIdx, the variable controlParaAct, and the array resSamplesRec when controlParaAct is equal to 2 as inputs, and the output is a modified reconstructed picture before deblocking filtering when controlParaAct is not equal to 1 or a modified residual sample array resSampleArray when controlParaAct is equal to 1.
5. The general decoding process for intra blocks as specified in this clause is invoked with the location (xTb1, yTb1), the variable log2TrafoSize set equal to log2TrafoSize – 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode predModeIntra, the variable cIdx, the variable controlParaAct, and the array resSamplesRec when controlParaAct is equal to 2 as inputs, and the output is a modified reconstructed picture before deblocking filtering when controlParaAct is not equal to 1 or a modified residual sample array resSampleArray when controlParaAct is equal to 1.
- Otherwise (splitFlag is equal to 0), for the variable blkIdx proceeding over the values 0..(cIdx > 0 && ChromaArrayType == 2 ? 1 : 0), the following ordered steps apply:
 1. The variable nTbS is set equal to $1 \ll \log_2 \text{TrafoSize}$.
 2. The variable yTbOffset is set equal to blkIdx * nTbS.
 3. The variable yTbOffsetY is set equal to yTbOffset * SubHeightC.
 4. When controlParaAct is not equal to 2, the variable residualDpcm is derived as follows:
 - If all of the following conditions are true, residualDpcm is set equal to 1.
 - implicit_rdpem_enabled_flag is equal to 1.
 - either transform_skip_flag[xTbY][yTbY + yTbOffsetY][cIdx] is equal to 1, or cu_transquant_bypass_flag is equal to 1.
 - either predModeIntra is equal to 10, or predModeIntra is equal to 26.
 - Otherwise, residualDpcm is set equal to 0.
 5. When controlParaAct is not equal to 1, the general intra sample prediction process as specified in clause 8.4.4.2.1 is invoked with the transform block location (xTb0, yTb0 + yTbOffset), the intra prediction mode predModeIntra, the transform block size nTbS and the variable cIdx as inputs, and the output is an (nTbS)x(nTbS) array predSamples.
 6. When controlParaAct is not equal to 2, the scaling and transformation process as specified in clause 8.6.2 is invoked with the luma location (xTbY, yTbY + yTbOffsetY), the variable trafoDepth, the variable cIdx and the transform size trafoSize set equal to nTbS as inputs, and the output is an (nTbS)x(nTbS) array resSamples.
 7. When controlParaAct is not equal to 2 and residualDpcm is equal to 1, the directional residual modification process for blocks using a transform bypass as specified in clause 8.6.5 is invoked with the variable mDir set equal to predModeIntra / 26, the variable nTbS and the (nTbS)x(nTbS) array r set equal to the array resSamples as inputs, and the output is a modified (nTbS)x(nTbS) array resSamples.
 8. When controlParaAct is not equal to 2 and cross_component_prediction_enabled_flag is equal to 1, ChromaArrayType is equal to 3, and cIdx is not equal to 0, the residual modification process for transform blocks using cross-component prediction as specified in clause 8.6.6 is invoked with the current luma transform block location (xTbY, yTbY), the variable nTbS, the variable cIdx, the (nTbS)x(nTbS) array r_Y set equal to the corresponding luma residual sample array resSamples of the current transform block and the (nTbS)x(nTbS) array r set equal to the array resSamples as inputs, and the output is a modified (nTbS)x(nTbS) array resSamples.
 9. When controlParaAct is not equal to 0, the following applies:
 - The variable nCbS specifying the size of the current coding block is derived by setting nCbS equal to $1 \ll (\log_2 \text{TrafoSize} + \text{trafDepth})$.
 - The sample location (xTbInCb, yTbInCb) specifying the top-left sample of the current transform block relative to the top-left sample of the current coding block is derived by setting xTbInCb equal to xTb0 & (nCbS – 1) and setting yTbInCb equal to yTb0 & (nCbS – 1).
 10. When controlParaAct is equal to 2, the (nTbS)x(nTbS) array resSamples is derived by setting resSamples[x][y] equal to resSamplesRec[x + xTbInCb][y + yTbInCb], for x and y in the range of 0 to nTbS – 1, inclusive.

11. The following applies:

- If controlParaAct is equal to 1, the residual array resSamplesArray is modified by setting resSamplesArray[x + xTbInCb][y + yTbInCb] equal to resSamples[x][y], for x and y in the range of 0 to nTbS – 1, inclusive.
- Otherwise (controlParaAct is not equal to 1), the picture construction process prior to in-loop filtering for a colour component as specified in clause 8.6.7 is invoked with the transform block location (xTb0, yTb0 + yTbOffset), the variables nCurrSw and nCurrSh both set equal to nTbS, the variable cIdx, the (nTbS)x(nTbS) array predSamples and the (nTbS)x(nTbS) array resSamples as inputs.

8.4.4.2 Intra sample prediction

8.4.4.2.1 General intra sample prediction

Inputs to this process are:

- a sample location (xTbCmp, yTbCmp) specifying the top-left sample of the current transform block relative to the top-left sample of the current picture,
- a variable predModeIntra specifying the intra prediction mode,
- a variable nTbS specifying the transform block size,
- a variable cIdx specifying the colour component of the current block.

Outputs of this process are the predicted samples predSamples[x][y], with x, y = 0..nTbS – 1.

The nTbS * 4 + 1 neighbouring samples p[x][y] that are constructed samples prior to the deblocking filter process, with x = -1, y = -1..nTbS * 2 – 1 and x = 0..nTbS * 2 – 1, y = -1, are derived as follows:

- The neighbouring location (xNbCmp, yNbCmp) is specified by:

$$(xNbCmp, yNbCmp) = (xTbCmp + x, yTbCmp + y) \quad (8-33)$$

- The current luma location (xTbY, yTbY) and the neighbouring luma location (xNbY, yNbY) are derived as follows:

$$(xTbY, yTbY) = (cIdx == 0) ? (xTbCmp, yTbCmp) : (xTbCmp * SubWidthC, yTbCmp * SubHeightC) \quad (8-34)$$

$$(xNbY, yNbY) = (cIdx == 0) ? (xNbCmp, yNbCmp) : (xNbCmp * SubWidthC, yNbCmp * SubHeightC) \quad (8-35)$$

- The availability derivation process for a block in z-scan order as specified in clause 6.4.1 is invoked with the current luma location (xCurr, yCurr) set equal to (xTbY, yTbY) and the neighbouring luma location (xNbY, yNbY) as inputs, and the output is assigned to availableN.
- Each sample p[x][y] is derived as follows:
 - If one or more of the following conditions are true, the sample p[x][y] is marked as "not available for intra prediction":
 - The variable availableN is equal to FALSE.
 - CuPredMode[xNbY][yNbY] is not equal to MODE_INTRA and constrained_intra_pred_flag is equal to 1.
 - Otherwise, the sample p[x][y] is marked as "available for intra prediction" and the sample at the location (xNbCmp, yNbCmp) is assigned to p[x][y].

When at least one sample p[x][y] with x = -1, y = -1..nTbS * 2 – 1 and x = 0..nTbS * 2 – 1, y = -1 is marked as "not available for intra prediction", the reference sample substitution process for intra sample prediction in clause 8.4.4.2.2 is invoked with the samples p[x][y] with x = -1, y = -1..nTbS * 2 – 1 and x = 0..nTbS * 2 – 1, y = -1, nTbS and cIdx as inputs, and the modified samples p[x][y] with x = -1, y = -1..nTbS * 2 – 1 and x = 0..nTbS * 2 – 1, y = -1 as output.

Depending on the value of predModeIntra, the following ordered steps apply:

1. When `intra_smoothing_disabled_flag` is equal to 0 and either `cIdx` is equal to 0 or `ChromaArrayType` is equal to 3, the filtering process of neighbouring samples specified in clause 8.4.4.2.3 is invoked with the sample array `p`, the transform block size `nTbS` and the colour component index `cIdx` as inputs, and the output is reassigned to the sample array `p`.
2. The intra sample prediction process according to `predModeIntra` applies as follows:
 - If `predModeIntra` is equal to `INTRA_PLANAR`, the corresponding intra prediction mode specified in clause 8.4.4.2.4 is invoked with the sample array `p` and the transform block size `nTbS` as inputs, and the output is the predicted sample array `predSamples`.
 - Otherwise, if `predModeIntra` is equal to `INTRA_DC`, the corresponding intra prediction mode specified in clause 8.4.4.2.5 is invoked with the sample array `p`, the transform block size `nTbS` and the colour component index `cIdx` as inputs, and the output is the predicted sample array `predSamples`.
 - Otherwise (`predModeIntra` is in the range of `INTRA_ANGULAR2..INTRA_ANGULAR34`), the corresponding intra prediction mode specified in clause 8.4.4.2.6 is invoked with the intra prediction mode `predModeIntra`, the sample array `p`, the transform block size `nTbS` and the colour component index `cIdx` as inputs, and the output is the predicted sample array `predSamples`.

8.4.4.2.2 Reference sample substitution process for intra sample prediction

Inputs to this process are:

- reference samples `p[x][y]` with $x = -1, y = -1..nTbS * 2 - 1$ and $x = 0..nTbS * 2 - 1, y = -1$ for intra sample prediction,
- a transform block size `nTbS`, and
- a variable `cIdx` specifying the colour component of the current block.

Outputs of this process are the modified reference samples `p[x][y]` with $x = -1, y = -1..nTbS * 2 - 1$ and $x = 0..nTbS * 2 - 1, y = -1$ for intra sample prediction.

The variable `bitDepth` is derived as follows:

- If `cIdx` is equal to 0, `bitDepth` is set equal to `BitDepthY`.
- Otherwise, `bitDepth` is set equal to `BitDepthC`.

The values of the samples `p[x][y]` with $x = -1, y = -1..nTbS * 2 - 1$ and $x = 0..nTbS * 2 - 1, y = -1$ are modified as follows:

- If all samples `p[x][y]` with $x = -1, y = -1..nTbS * 2 - 1$ and $x = 0..nTbS * 2 - 1, y = -1$ are marked as "not available for intra prediction", the value $1 \ll (\text{bitDepth} - 1)$ is substituted for the values of all samples `p[x][y]`.
- Otherwise (at least one but not all samples `p[x][y]` are marked as "not available for intra prediction"), the following ordered steps are performed:
 1. When `p[-1][nTbS * 2 - 1]` is marked as "not available for intra prediction", search sequentially starting from $x = -1, y = nTbS * 2 - 1$ to $x = -1, y = -1$, then from $x = 0, y = -1$ to $x = nTbS * 2 - 1, y = -1$. Once a sample `p[x][y]` marked as "available for intra prediction" is found, the search is terminated and the value of `p[x][y]` is assigned to `p[-1][nTbS * 2 - 1]`.
 2. Search sequentially starting from $x = -1, y = nTbS * 2 - 2$ to $x = -1, y = -1$, when `p[x][y]` is marked as "not available for intra prediction", the value of `p[x][y + 1]` is substituted for the value of `p[x][y]`.
 3. For $x = 0..nTbS * 2 - 1, y = -1$, when `p[x][y]` is marked as "not available for intra prediction", the value of `p[x - 1][y]` is substituted for the value of `p[x][y]`.

All samples `p[x][y]` with $x = -1, y = -1..nTbS * 2 - 1$ and $x = 0..nTbS * 2 - 1, y = -1$ are marked as "available for intra prediction".

8.4.4.2.3 Filtering process of neighbouring samples

Inputs to this process are:

- the neighbouring samples `p[x][y]`, with $x = -1, y = -1..nTbS * 2 - 1$ and $x = 0..nTbS * 2 - 1, y = -1$,
- a variable `nTbS` specifying the transform block size.

Outputs of this process are the filtered samples `pF[x][y]`, with $x = -1, y = -1..nTbS * 2 - 1$ and $x = 0..nTbS * 2 - 1, y = -1$.

The variable filterFlag is derived as follows:

- If one or more of the following conditions are true, filterFlag is set equal to 0:
 - predModeIntra is equal to INTRA_DC.
 - nTbS is equal to 4.
- Otherwise, the following applies:
 - The variable minDistVerHor is set equal to $\text{Min}(\text{Abs}(\text{predModeIntra} - 26), \text{Abs}(\text{predModeIntra} - 10))$.
 - The variable intraHorVerDistThres[nTbS] is specified in Table 8-4.
 - The variable filterFlag is derived as follows:
 - If minDistVerHor is greater than intraHorVerDistThres[nTbS], filterFlag is set equal to 1.
 - Otherwise, filterFlag is set equal to 0.

Table 8-4 – Specification of intraHorVerDistThres[nTbS] for various transform block sizes

	nTbS = 8	nTbS = 16	nTbS = 32
intraHorVerDistThres[nTbS]	7	1	0

When filterFlag is equal to 1, the following applies:

- The variable biIntFlag is derived as follows:
 - If all of the following conditions are true, biIntFlag is set equal to 1:
 - strong_intra_smoothing_enabled_flag is equal to 1.
 - cIdx is equal to 0.
 - nTbS is equal to 32.
 - $\text{Abs}(p[-1][-1] + p[\text{nTbS} * 2 - 1][-1] - 2 * p[\text{nTbS} - 1][-1]) \text{ is less than } 1 \ll (\text{BitDepth}_Y - 5)$.
 - $\text{Abs}(p[-1][-1] + p[-1][\text{nTbS} * 2 - 1] - 2 * p[-1][\text{nTbS} - 1]) \text{ is less than } 1 \ll (\text{BitDepth}_Y - 5)$.
 - Otherwise, biIntFlag is set equal to 0.
- The filtering is performed as follows:
 - If biIntFlag is equal to 1, the filtered sample values pF[x][y] with $x = -1, y = -1..63$ and $x = 0..63, y = -1$ are derived as follows:

$$pF[-1][-1] = p[-1][-1] \quad (8-36)$$

$$pF[-1][y] = ((63 - y) * p[-1][-1] + (y + 1) * p[-1][63] + 32) \gg 6 \text{ for } y = 0..62 \quad (8-37)$$

$$pF[-1][63] = p[-1][63] \quad (8-38)$$

$$pF[x][-1] = ((63 - x) * p[-1][-1] + (x + 1) * p[63][-1] + 32) \gg 6 \text{ for } x = 0..62 \quad (8-39)$$

$$pF[63][-1] = p[63][-1] \quad (8-40)$$

- Otherwise (biIntFlag is equal to 0), the filtered sample values pF[x][y] with $x = -1, y = -1..nTbS * 2 - 1$ and $x = 0..nTbS * 2 - 1, y = -1$ are derived as follows:

$$pF[-1][-1] = (p[-1][0] + 2 * p[-1][-1] + p[0][-1] + 2) \gg 2 \quad (8-41)$$

$$pF[-1][y] = (p[-1][y + 1] + 2 * p[-1][y] + p[-1][y - 1] + 2) \gg 2 \text{ for } y = 0..nTbS * 2 - 2 \quad (8-42)$$

$$pF[-1][\text{nTbS} * 2 - 1] = p[-1][\text{nTbS} * 2 - 1] \quad (8-43)$$

$$pF[x][y] = (p[x-1][y] + 2 * p[x][y-1] + p[x+1][y] + 2) >> 2 \text{ for } x = 0..nTbS * 2 - 2 \quad (8-44)$$

$$pF[nTbS * 2 - 1][y] = p[nTbS * 2 - 1][y-1] \quad (8-45)$$

8.4.4.2.4 Specification of intra prediction mode INTRA_PLANAR

Inputs to this process are:

- the neighbouring samples $p[x][y]$, with $x = -1, y = -1..nTbS * 2 - 1$ and $x = 0..nTbS * 2 - 1, y = -1$,
- a variable $nTbS$ specifying the transform block size.

Outputs of this process are the predicted samples $predSamples[x][y]$, with $x, y = 0..nTbS - 1$.

The values of the prediction samples $predSamples[x][y]$, with $x, y = 0..nTbS - 1$, are derived as follows:

$$predSamples[x][y] = ((nTbS - 1 - x) * p[-1][y] + (x + 1) * p[nTbS][y-1] + (nTbS - 1 - y) * p[x][y-1] + (y + 1) * p[-1][nTbS] + nTbS) >> (Log2(nTbS) + 1) \quad (8-46)$$

8.4.4.2.5 Specification of intra prediction mode INTRA_DC

Inputs to this process are:

- the neighbouring samples $p[x][y]$, with $x = -1, y = -1..nTbS * 2 - 1$ and $x = 0..nTbS * 2 - 1, y = -1$,
- a variable $nTbS$ specifying the transform block size,
- a variable $cIdx$ specifying the colour component of the current block.

Outputs of this process are the predicted samples $predSamples[x][y]$, with $x, y = 0..nTbS - 1$.

The values of the prediction samples $predSamples[x][y]$, with $x, y = 0..nTbS - 1$, are derived by the following ordered steps:

1. A variable $dcVal$ is derived as follows:

$$dcVal = \left(\sum_{x'=0}^{nTbS-1} p[x'][y-1] + \sum_{y'=0}^{nTbS-1} p[-1][y'] + nTbS \right) >> (k + 1) \quad (8-47)$$

where $k = Log2(nTbS)$.

2. Depending on the value of the colour component index $cIdx$, the following applies:

- If $cIdx$ is equal to 0, $intra_boundary_filtering_disabled_flag$ is equal to 0, and $nTbS$ is less than 32, the following applies:

$$predSamples[0][0] = (p[-1][0] + 2 * dcVal + p[0][y-1] + 2) >> 2 \quad (8-48)$$

$$predSamples[x][0] = (p[x][y-1] + 3 * dcVal + 2) >> 2, \text{ with } x = 1..nTbS - 1 \quad (8-49)$$

$$predSamples[0][y] = (p[-1][y] + 3 * dcVal + 2) >> 2, \text{ with } y = 1..nTbS - 1 \quad (8-50)$$

$$predSamples[x][y] = dcVal, \text{ with } x, y = 1..nTbS - 1 \quad (8-51)$$

- Otherwise, the prediction samples $predSamples[x][y]$ are derived as follows:

$$predSamples[x][y] = dcVal, \text{ with } x, y = 0..nTbS - 1 \quad (8-52)$$

8.4.4.2.6 Specification of intra prediction mode in the range of INTRA_ANGULAR2.. INTRA_ANGULAR34

Inputs to this process are:

- the intra prediction mode $predModeIntra$,

- the neighbouring samples $p[x][y]$, with $x = -1, y = -1..nTbS * 2 - 1$ and $x = 0..nTbS * 2 - 1, y = -1$,
- a variable $nTbS$ specifying the transform block size,
- a variable $cIdx$ specifying the colour component of the current block.

Outputs of this process are the predicted samples $predSamples[x][y]$, with $x, y = 0..nTbS - 1$.

Figure 8-2 illustrates the total 33 intra angles and Table 8-5 specifies the mapping table between $predModeIntra$ and the angle parameter $intraPredAngle$.

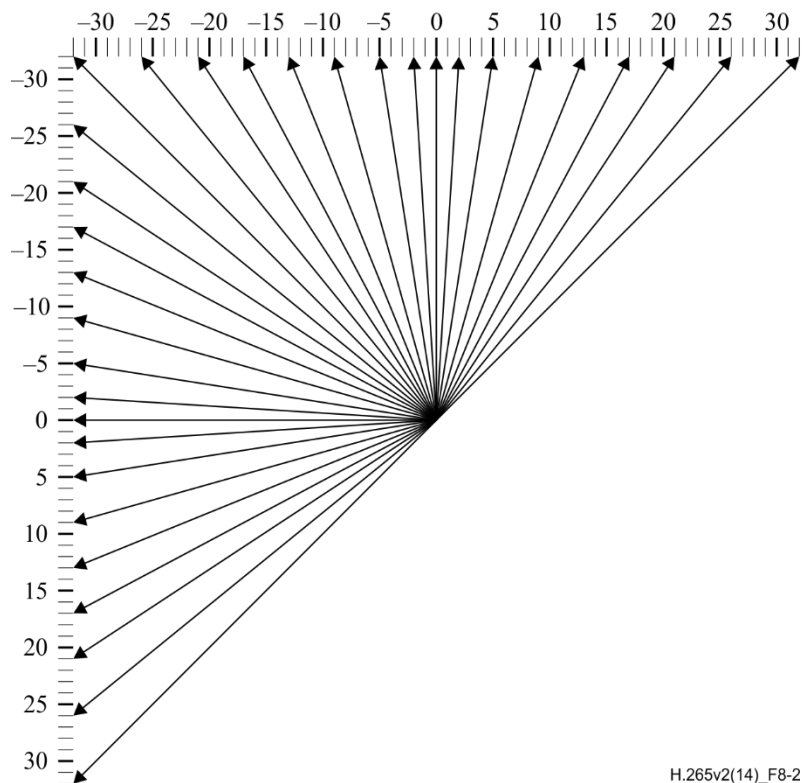


Figure 8-2 – Intra prediction angle definition (informative)

Table 8-5 – Specification of $intraPredAngle$

predModeIntra	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
intraPredAngle	-	32	26	21	17	13	9	5	2	0	-2	-5	-9	-13	-17	-21	-26
predModeIntra	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
intraPredAngle	-32	-26	-21	-17	-13	-9	-5	-2	0	2	5	9	13	17	21	26	32

Table 8-6 further specifies the mapping table between $predModeIntra$ and the inverse angle parameter $invAngle$.

Table 8-6 – Specification of $invAngle$

predModeIntra	11	12	13	14	15	16	17	18
invAngle	-4 096	-1 638	-910	-630	-482	-390	-315	-256
predModeIntra	19	20	21	22	23	24	25	26
invAngle	-315	-390	-482	-630	-910	-1 638	-4 096	-

The variable `disableIntraBoundaryFilter` is derived as follows:

- If `intra_boundary_filtering_disabled_flag` is equal to 1, `disableIntraBoundaryFilter` is set equal to 1.
- Otherwise (`intra_boundary_filtering_disabled_flag` is equal to 0), if `implicit_rdpem_enabled_flag` and `cu_transquant_bypass_flag` are both equal to 1, `disableIntraBoundaryFilter` is set equal to 1.
- Otherwise, `disableIntraBoundaryFilter` is set equal to 0.

The values of the prediction samples `predSamples[x][y]`, with $x, y = 0..nTbS - 1$ are derived as follows:

- If `predModeIntra` is greater than or equal to 18, the following ordered steps apply:

1. The reference sample array `ref[x]` is specified as follows:

- The following applies:

$$\text{ref}[x] = p[-1 + x][-1], \text{ with } x = 0..nTbS \quad (8-53)$$

- If `intraPredAngle` is less than 0, the main reference sample array is extended as follows:

- When $(nTbS * \text{intraPredAngle}) \gg 5$ is less than -1 ,

$$\begin{aligned} \text{ref}[x] &= p[-1][-1 + ((x * \text{invAngle} + 128) \gg 8)], \\ &\text{with } x = -1..(nTbS * \text{intraPredAngle}) \gg 5 \end{aligned} \quad (8-54)$$

- Otherwise,

$$\text{ref}[x] = p[-1 + x][-1], \text{ with } x = nTbS + 1..2 * nTbS \quad (8-55)$$

2. The values of the prediction samples `predSamples[x][y]`, with $x, y = 0..nTbS - 1$ are derived as follows:

- a. The index variable `iIdx` and the multiplication factor `iFact` are derived as follows:

$$\text{iIdx} = ((y + 1) * \text{intraPredAngle}) \gg 5 \quad (8-56)$$

$$\text{iFact} = ((y + 1) * \text{intraPredAngle}) \& 31 \quad (8-57)$$

- b. Depending on the value of `iFact`, the following applies:

- If `iFact` is not equal to 0, the value of the prediction samples `predSamples[x][y]` is derived as follows:

$$\begin{aligned} \text{predSamples}[x][y] &= \\ &((32 - \text{iFact}) * \text{ref}[x + \text{iIdx} + 1] + \text{iFact} * \text{ref}[x + \text{iIdx} + 2] + 16) \gg 5 \end{aligned} \quad (8-58)$$

- Otherwise, the value of the prediction samples `predSamples[x][y]` is derived as follows:

$$\text{predSamples}[x][y] = \text{ref}[x + \text{iIdx} + 1] \quad (8-59)$$

- c. When `predModeIntra` is equal to 26 (vertical), `cIdx` is equal to 0, `nTbS` is less than 32, and `disableIntraBoundaryFilter` is equal to 0, the following filtering applies with $x = 0, y = 0..nTbS - 1$:

$$\text{predSamples}[x][y] = \text{Clip1}_V(p[x][-1] + ((p[-1][y] - p[-1][-1]) \gg 1)) \quad (8-60)$$

- Otherwise (`predModeIntra` is less than 18), the following ordered steps apply:

1. The reference sample array `ref[x]` is specified as follows:

- The following applies:

$$\text{ref}[x] = p[-1][-1 + x], \text{ with } x = 0..nTbS \quad (8-61)$$

- If `intraPredAngle` is less than 0, the main reference sample array is extended as follows:

- When $(nTbS * \text{intraPredAngle}) \gg 5$ is less than -1 ,

$$\begin{aligned} \text{ref}[x] &= p[-1 + ((x * \text{invAngle} + 128) \gg 8)][-1], \\ &\text{with } x = -1..(nTbS * \text{intraPredAngle}) \gg 5 \end{aligned} \quad (8-62)$$

– Otherwise,

$$\text{ref}[x] = p[-1][-1 + x], \text{ with } x = nTbS + 1..2 * nTbS \quad (8-63)$$

2. The values of the prediction samples $\text{predSamples}[x][y]$, with $x, y = 0..nTbS - 1$ are derived as follows:

a. The index variable $iIdx$ and the multiplication factor $iFact$ are derived as follows:

$$iIdx = ((x + 1) * \text{intraPredAngle}) \gg 5 \quad (8-64)$$

$$iFact = ((x + 1) * \text{intraPredAngle}) \& 31 \quad (8-65)$$

b. Depending on the value of $iFact$, the following applies:

– If $iFact$ is not equal to 0, the value of the prediction samples $\text{predSamples}[x][y]$ is derived as follows:

$$\begin{aligned} \text{predSamples}[x][y] &= \\ &((32 - iFact) * \text{ref}[y + iIdx + 1] + iFact * \text{ref}[y + iIdx + 2] + 16) \gg 5 \end{aligned} \quad (8-66)$$

– Otherwise, the value of the prediction samples $\text{predSamples}[x][y]$ is derived as follows:

$$\text{predSamples}[x][y] = \text{ref}[y + iIdx + 1] \quad (8-67)$$

c. When predModeIntra is equal to 10 (horizontal), $cIdx$ is equal to 0, $nTbS$ is less than 32 and $\text{disableIntraBoundaryFilter}$ is equal to 0, the following filtering applies with $x = 0..nTbS - 1, y = 0$:

$$\text{predSamples}[x][y] = \text{Clip1}_Y(p[-1][y] + ((p[x][-1] - p[-1][-1]) \gg 1)) \quad (8-68)$$

8.4.4.2.7 Decoding process for palette mode

Inputs to this process are:

- a location (xCb, yCb) specifying the top-left luma sample of the current block relative to the top-left luma sample of the current picture,
- a variable $cIdx$ specifying the colour component of the current block,
- two variables $nCbSX$ and $nCbSY$ specifying the width and height of the current block, respectively.

Output of this process is an array $\text{recSamples}[x][y]$, with $x = 0..nCbSX - 1, y = 0..nCbSY - 1$ specifying reconstructed sample values for the block.

Depending on the value of $cIdx$, the variables $nSubWidth$ and $nSubHeight$ are derived as follows:

- If $cIdx$ is equal to 0, $nSubWidth$ is set to 1 and $nSubHeight$ is set to 1.
- Otherwise, $nSubWidth$ is set to SubWidthC and $nSubHeight$ is set to SubHeightC .

The $(nCbSX \times nCbSY)$ block of the reconstructed sample array recSamples at location (xCb, yCb) is represented by $\text{recSamples}[x][y]$ with $x = 0..nCbSX - 1$ and $y = 0..nCbSY - 1$, and the value of $\text{recSamples}[x][y]$ for each x in the range of 0 to $nCbSX - 1$, inclusive, and each y in the range of 0 to $nCbSY - 1$, inclusive, is derived as follows:

– The variables xL and yL are derived as follows:

$$xL = \text{palette_transpose_flag} ? y * nSubHeight : x * nSubWidth \quad (8-69)$$

$$yL = \text{palette_transpose_flag} ? x * nSubWidth : y * nSubHeight \quad (8-70)$$

– The variable $bIsEscapeSample$ is derived as follows:

- If $\text{PaletteIndexMap}[x_{Cb} + x_L][y_{Cb} + y_L]$ is equal to MaxPaletteIndex and $\text{palette_escape_val_present_flag}$ is equal to 1, bIsEscapeSample is set equal to 1.
- Otherwise, bIsEscapeSample is set equal to 0.
- If bIsEscapeSample is equal to 0, the following applies:

$$\text{recSamples}[x][y] = \text{CurrentPaletteEntries}[cIdx][\text{PaletteIndexMap}[x_{Cb} + x_L][y_{Cb} + y_L]] \quad (8-71)$$

- Otherwise, if $\text{cu_transquant_bypass_flag}$ is equal to 1, the following applies:

$$\text{recSamples}[x][y] = \text{PaletteEscapeVal}[cIdx][x_{Cb} + x_L][y_{Cb} + y_L] \quad (8-72)$$

- Otherwise (bIsEscapeSample is equal to 1 and $\text{cu_transquant_bypass_flag}$ is equal to 0), the following ordered steps apply:
 1. The derivation process for quantization parameters as specified in clause 8.6.1 is invoked with the location (x_{Cb}, y_{Cb}) specifying the top-left sample of the current block relative to the top-left sample of the current picture.
 2. The quantization parameter qP is derived as follows:
 - If $cIdx$ is equal to 0,

$$qP = \text{Max}(0, Qp'Y) \quad (8-73)$$

- Otherwise, if $cIdx$ is equal to 1,

$$qP = \text{Max}(0, Qp'Cb) \quad (8-74)$$

- Otherwise ($cIdx$ is equal to 2),

$$qP = \text{Max}(0, Qp'Cr) \quad (8-75)$$

3. The variable bitDepth is derived as follows:

$$\text{bitDepth} = (cIdx == 0) ? \text{BitDepth}_Y : \text{BitDepth}_C \quad (8-76)$$

4. The list $\text{levelScale}[]$ is specified as $\text{levelScale}[k] = \{40, 45, 51, 57, 64, 72\}$ with $k = 0..5$.

5. The following applies:

$$\text{tmpVal} = (\text{PaletteEscapeVal}[cIdx][x_{Cb} + x_L][y_{Cb} + y_L] * \text{levelScale}[qP \% 6]) \ll ((qP / 6) + 32) \gg 6 \quad (8-77)$$

$$\text{recSamples}[x][y] = \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, \text{tmpVal}) \quad (8-78)$$

The variable $\text{PredictorPaletteSize}$ and the array $\text{PredictorPaletteEntries}$ are derived or modified as follows:

```

numComps = ( ChromaArrayType == 0 ) ? 1 : 3
for( i = 0; i < CurrentPaletteSize; i++ )
    for( cIdx = 0; cIdx < numComps; cIdx++ )
        newPredictorPaletteEntries[ cIdx ][ i ] = CurrentPaletteEntries[ cIdx ][ i ]
newPredictorPaletteSize = CurrentPaletteSize
for( i = 0; i < PredictorPaletteSize && newPredictorPaletteSize < PaletteMaxPredictorSize; i++ )
    if( !PalettePredictorEntryReuseFlags[ i ] ) {
        for( cIdx = 0; cIdx < numComps; cIdx++ )
            newPredictorPaletteEntries[ cIdx ][ newPredictorPaletteSize ] =
                PredictorPaletteEntries[ cIdx ][ i ]
        newPredictorPaletteSize++
    }
for( cIdx = 0; cIdx < numComps; cIdx++ )

```

```

for( i = 0; i < newPredictorPaletteSize; i++ )
    PredictorPaletteEntries[ cIdx ][ i ] = newPredictorPaletteEntries[ cIdx ][ i ]
PredictorPaletteSize = newPredictorPaletteSize

```

It is a requirement of bitstream conformance that the value of PredictorPaletteSize shall be in the range of 0 to PaletteMaxPredictorSize, inclusive.

8.5 Decoding process for coding units coded in inter prediction mode

8.5.1 General decoding process for coding units coded in inter prediction mode

Inputs to this process are:

- a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a variable log2CbSize specifying the size of the current coding block.

Output of this process is a modified reconstructed picture before deblocking filtering.

The derivation process for quantization parameters as specified in clause 8.6.1 is invoked with the luma location (xCb, yCb) as input.

The variable nCbS_L is set equal to $1 \ll \log_2 \text{CbSize}$. When ChromaArrayType is not equal to 0, the variable nCbSw_C is set equal to $(1 \ll \log_2 \text{CbSize}) / \text{SubWidthC}$ and the variable nCbSh_C is set equal to $(1 \ll \log_2 \text{CbSize}) / \text{SubHeightC}$.

The decoding process for coding units coded in inter prediction mode consists of the following ordered steps:

1. The inter prediction process as specified in clause 8.5.2 is invoked with the luma location (xCb, yCb) and the luma coding block size log2CbSize as inputs, and the outputs are the array predSamples_L and, when ChromaArrayType is not equal to 0, the arrays predSamples_{Cb} and predSamples_{Cr}.
2. The decoding process for the residual signal of coding units coded in inter prediction mode specified in clause 8.5.4 is invoked with the luma location (xCb, yCb) and the luma coding block size log2CbSize as inputs, and the outputs are the array resSamples_L and, when ChromaArrayType is not equal to 0, the arrays resSamples_{Cb} and resSamples_{Cr}.
3. The reconstructed samples of the current coding unit are derived as follows:
 - The picture construction process prior to in-loop filtering for a colour component as specified in clause 8.6.7 is invoked with the luma coding block location (xCb, yCb), the variable nCurrSw set equal to nCbS_L, the variable nCurrSh set equal to nCbS_L, the variable cIdx set equal to 0, the (nCbS_L)x(nCbS_L) array predSamples set equal to predSamples_L and the (nCbS_L)x(nCbS_L) array resSamples set equal to resSamples_L as inputs.
 - When ChromaArrayType is not equal to 0, the picture construction process prior to in-loop filtering for a colour component as specified in clause 8.6.7 is invoked with the chroma coding block location (xCb / SubWidthC, yCb / SubHeightC), the variable nCurrSw set equal to nCbSw_C, the variable nCurrSh set equal to nCbSh_C, the variable cIdx set equal to 1, the (nCbSw_C)x(nCbSh_C) array predSamples set equal to predSamples_{Cb} and the (nCbSw_C)x(nCbSh_C) array resSamples set equal to resSamples_{Cb} as inputs.
 - When ChromaArrayType is not equal to 0, the picture construction process prior to in-loop filtering for a colour component as specified in clause 8.6.7 is invoked with the chroma coding block location (xCb / SubWidthC, yCb / SubHeightC), the variable nCurrSw set equal to nCbSw_C, the variable nCurrSh set equal to nCbSh_C, the variable cIdx set equal to 2, the (nCbSw_C)x(nCbSh_C) array predSamples set equal to predSamples_{Cr} and the (nCbSw_C)x(nCbSh_C) array resSamples set equal to resSamples_{Cr} as inputs.

8.5.2 Inter prediction process

This process is invoked when decoding coding unit whose CuPredMode[xCb][yCb] is not equal to MODE_INTRA.

Inputs to this process are:

- a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a variable log2CbSize specifying the size of the current luma coding block.

Outputs of this process are:

- an (nCbS_L)x(nCbS_L) array predSamples_L of luma prediction samples, where nCbS_L is derived as specified below,

- when ChromaArrayType is not equal to 0, an $(nCbSw_C) \times (nCbSh_C)$ array $predSamples_{Cb}$ of chroma prediction samples for the component Cb, where $nCbSw_C$ and $nCbSh_C$ are derived as specified below,
- when ChromaArrayType is not equal to 0, an $(nCbSw_C) \times (nCbSh_C)$ array $predSamples_{Cr}$ of chroma prediction samples for the component Cr, where $nCbSw_C$ and $nCbSh_C$ are derived as specified below.

The variable $nCbS_L$ is set equal to $1 \ll \log_2 CbSize$. When ChromaArrayType is not equal to 0, the variable $nCbSw_C$ is set equal to $nCbS_L / SubWidthC$ and the variable $nCbSh_C$ is set equal to $nCbS_L / SubHeightC$.

The variable $nCbS_{1L}$ is set equal to $nCbS_L \gg 1$.

Depending on the value of PartMode, the following applies:

- If PartMode is equal to PART_2Nx2N, the decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBI, yBI) set equal to (0, 0), the size of the luma coding block $nCbS_L$, the width of the luma prediction block $nPbW$ set equal to $nCbS_L$, the height of the luma prediction block $nPbH$ set equal to $nCbS_L$ and a partition index $partIdx$ set equal to 0 as inputs, and the outputs are an $(nCbS_L) \times (nCbS_L)$ array $predSamples_L$ and, when ChromaArrayType is not equal to 0, two $(nCbSw_C) \times (nCbSh_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.
- Otherwise, if PartMode is equal to PART_2NxN, the following ordered steps apply:
 1. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBI, yBI) set equal to (0, 0), the size of the luma coding block $nCbS_L$, the width of the luma prediction block $nPbW$ set equal to $nCbS_L$, the height of the luma prediction block $nPbH$ set equal to $nCbS_L \gg 1$ and a partition index $partIdx$ set equal to 0 as inputs, and the outputs are an $(nCbS_L) \times (nCbS_L)$ array $predSamples_L$ and, when ChromaArrayType is not equal to 0, two $(nCbSw_C) \times (nCbSh_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.
 2. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBI, yBI) set equal to (0, $nCbS_L \gg 1$), the size of the luma coding block $nCbS_L$, the width of the luma prediction block $nPbW$ set equal to $nCbS_L$, the height of the luma prediction block $nPbH$ set equal to $nCbS_L \gg 1$ and a partition index $partIdx$ set equal to 1 as inputs, and the outputs are the modified $(nCbS_L) \times (nCbS_L)$ array $predSamples_L$ and, when ChromaArrayType is not equal to 0, the two modified $(nCbSw_C) \times (nCbSh_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.
- Otherwise, if PartMode is equal to PART_Nx2N, the following ordered steps apply:
 1. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBI, yBI) set equal to (0, 0), the size of the luma coding block $nCbS_L$, the width of the luma prediction block $nPbW$ set equal to $nCbS_L \gg 1$, the height of the luma prediction block $nPbH$ set equal to $nCbS_L$ and a partition index $partIdx$ set equal to 0 as inputs, and the outputs are an $(nCbS_L) \times (nCbS_L)$ array $predSamples_L$ and, when ChromaArrayType is not equal to 0, two $(nCbSw_C) \times (nCbSh_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.
 2. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBI, yBI) set equal to ($nCbS_L \gg 1$, 0), the size of the luma coding block $nCbS_L$, the width of the luma prediction block $nPbW$ set equal to $nCbS_L \gg 1$, the height of the luma prediction block $nPbH$ set equal to $nCbS_L$ and a partition index $partIdx$ set equal to 1 as inputs, and the outputs are the modified $(nCbS_L) \times (nCbS_L)$ array $predSamples_L$ and, when ChromaArrayType is not equal to 0, the two modified $(nCbSw_C) \times (nCbSh_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.
- Otherwise, if PartMode is equal to PART_2NxN_U, the following ordered steps apply:
 1. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBI, yBI) set equal to (0, 0), the size of the luma coding block $nCbS_L$, the width of the luma prediction block $nPbW$ set equal to $nCbS_L$, the height of the luma prediction block $nPbH$ set equal to $nCbS_L \gg 2$ and a partition index $partIdx$ set equal to 0 as inputs, and the outputs are an $(nCbS_L) \times (nCbS_L)$ array $predSamples_L$ and, when ChromaArrayType is not equal to 0, two $(nCbSw_C) \times (nCbSh_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.
 2. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (xCb, yCb), the luma location (xBI, yBI) set equal to (0, $nCbS_L \gg 2$), the size of the luma coding block $nCbS_L$, the width of the luma prediction block $nPbW$ set equal to $nCbS_L$, the height of the luma prediction block $nPbH$ set equal to $(nCbS_L \gg 1) + (nCbS_L \gg 2)$ and a partition index $partIdx$ set equal to 1 as inputs, and the outputs are the modified $(nCbS_L) \times (nCbS_L)$ array $predSamples_L$ and, when ChromaArrayType is not equal to 0, the two modified $(nCbSw_C) \times (nCbSh_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.
- Otherwise, if PartMode is equal to PART_2NxND, the following ordered steps apply:

- [illegible]

4. The decoding process for prediction units in inter prediction mode as specified in clause 8.5.3 is invoked with the luma location (x_{Cb} , y_{Cb}), the luma location (x_{Bl} , y_{Bl}) set equal to ($n_{CbS_L} \gg 1$, $n_{CbS_L} \gg 1$), the size of the luma coding block n_{CbS_L} , the width of the luma prediction block n_{PbW} set equal to $n_{CbS_L} \gg 1$, the height of the luma prediction block n_{PbH} set equal to $n_{CbS_L} \gg 1$ and a partition index $partIdx$ set equal to 3 as inputs, and the outputs are the modified $(n_{CbS_L}) \times (n_{CbS_L})$ array $predSamples_L$ and, when $ChromaArrayType$ is not equal to 0, the two modified $(n_{CbSw_C}) \times (n_{CbSh_C})$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

8.5.3 Decoding process for prediction units in inter prediction mode

8.5.3.1 General

Inputs to this process are:

- a luma location (x_{Cb} , y_{Cb}) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a luma location (x_{Bl} , y_{Bl}) specifying the top-left sample of the current luma prediction block relative to the top-left sample of the current luma coding block,
- a variable n_{CbS} specifying the size of the current luma coding block,
- a variable n_{PbW} specifying the width of the current luma prediction block,
- a variable n_{PbH} specifying the height of the current luma prediction block,
- a variable $partIdx$ specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

- an $(n_{CbS_L}) \times (n_{CbS_L})$ array $predSamples_L$ of luma prediction samples, where n_{CbS_L} is derived as specified below,
- when $ChromaArrayType$ is not equal to 0, an $(n_{CbSw_C}) \times (n_{CbSh_C})$ array $predSamples_{Cb}$ of chroma prediction samples for the component Cb, where n_{CbSw_C} and n_{CbSh_C} are derived as specified below,
- when $ChromaArrayType$ is not equal to 0, an $(n_{CbSw_C}) \times (n_{CbSh_C})$ array $predSamples_{Cr}$ of chroma prediction samples for the component Cr, where n_{CbSw_C} and n_{CbSh_C} are derived as specified below.

The variable n_{CbS_L} is set equal to n_{CbS} . When $ChromaArrayType$ is not equal to 0, the variable n_{CbSw_C} is set equal to $n_{CbS} / SubWidthC$ and the variable n_{CbSh_C} is set equal to $n_{CbS} / SubHeightC$.

The decoding process for prediction units in inter prediction mode consists of the following ordered steps:

1. The derivation process for motion vector components and reference indices as specified in clause 8.5.3.2 is invoked with the luma coding block location (x_{Cb} , y_{Cb}), the luma prediction block location (x_{Bl} , y_{Bl}), the luma coding block size block n_{CbS} , the luma prediction block width n_{PbW} , the luma prediction block height n_{PbH} and the prediction unit index $partIdx$ as inputs, and the luma motion vectors $mvL0$ and $mvL1$, when $ChromaArrayType$ is not equal to 0, the chroma motion vectors $mvCL0$ and $mvCL1$, the reference indices $refIdxL0$ and $refIdxL1$ and the prediction list utilization flags $predFlagL0$ and $predFlagL1$ as outputs.
2. The decoding process for inter sample prediction as specified in clause 8.5.3.3 is invoked with the luma coding block location (x_{Cb} , y_{Cb}), the luma prediction block location (x_{Bl} , y_{Bl}), the luma coding block size block n_{CbS} , the luma prediction block width n_{PbW} , the luma prediction block height n_{PbH} , the luma motion vectors $mvL0$ and $mvL1$, when $ChromaArrayType$ is not equal to 0, the chroma motion vectors $mvCL0$ and $mvCL1$, the reference indices $refIdxL0$ and $refIdxL1$, and the prediction list utilization flags $predFlagL0$ and $predFlagL1$ as inputs, and the inter prediction samples ($predSamples$) that are an $(n_{CbS_L}) \times (n_{CbS_L})$ array $predSamples_L$ of prediction luma samples and, when $ChromaArrayType$ is not equal to 0, two $(n_{CbSw_C}) \times (n_{CbSh_C})$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$ of prediction chroma samples, one for each of the chroma components Cb and Cr, as outputs.

For use in derivation processes of variables invoked later in the decoding process, the following assignments are made for $x = x_{Bl}..x_{Bl} + n_{PbW} - 1$ and $y = y_{Bl}..y_{Bl} + n_{PbH} - 1$:

$$MvL0[x_{Cb} + x][y_{Cb} + y] = mvL0 \quad (8-80)$$

$$MvL1[x_{Cb} + x][y_{Cb} + y] = mvL1 \quad (8-81)$$

$$RefIdxL0[x_{Cb} + x][y_{Cb} + y] = refIdxL0 \quad (8-82)$$

$$RefIdxL1[x_{Cb} + x][y_{Cb} + y] = refIdxL1 \quad (8-83)$$

$$\text{PredFlagL0}[x_{Cb} + x][y_{Cb} + y] = \text{predFlagL0} \quad (8-84)$$

$$\text{PredFlagL1}[x_{Cb} + x][y_{Cb} + y] = \text{predFlagL1} \quad (8-85)$$

8.5.3.2 Derivation process for motion vector components and reference indices

8.5.3.2.1 General

Inputs to this process are:

- a luma location (x_{Cb} , y_{Cb}) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a luma location (x_{Bl} , y_{Bl}) of the top-left sample of the current luma prediction block relative to the top-left sample of the current luma coding block,
- a variable n_{CbS} specifying the size of the current luma coding block,
- two variables n_{PbW} and n_{PbH} specifying the width and the height of the luma prediction block,
- a variable $partIdx$ specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

- the luma motion vectors $mvL0$ and $mvL1$,
- when $ChromaArrayType$ is not equal to 0, the chroma motion vectors $mvCL0$ and $mvCL1$,
- the reference indices $refIdxL0$ and $refIdxL1$,
- the prediction list utilization flags $predFlagL0$ and $predFlagL1$.

Let (x_{Pb} , y_{Pb}) specify the top-left sample location of the current luma prediction block relative to the top-left luma sample of the current picture where $x_{Pb} = x_{Cb} + x_{Bl}$ and $y_{Pb} = y_{Cb} + y_{Bl}$.

Let the variable LX be $RefPicListX$, with X being 0 or 1, of the current picture.

The variables n_{PbSw} , and n_{PbSh} are derived as follows:

$$n_{PbSw} = n_{CbS} / ((PartMode == PART_2Nx2N \parallel PartMode == PART_2NxN) ? 1 : 2) \quad (8-86)$$

$$n_{PbSh} = n_{CbS} / ((PartMode == PART_2Nx2N \parallel PartMode == PART_Nx2N) ? 1 : 2) \quad (8-87)$$

The function $LongTermRefPic(aPic, aPb, refIdx, LX)$, with X being 0 or 1, is defined as follows:

- If the picture with index $refIdx$ from reference picture list LX of the slice containing prediction block aPb in the picture $aPic$ was marked as "used for long-term reference" at the time when $aPic$ was the current picture, $LongTermRefPic(aPic, aPb, refIdx, LX)$ is equal to 1.
- Otherwise, $LongTermRefPic(aPic, aPb, refIdx, LX)$ is equal to 0.

For the derivation of the variables $mvL0$ and $mvL1$, $refIdxL0$ and $refIdxL1$, as well as $predFlagL0$ and $predFlagL1$, the following applies:

- If $merge_flag[x_{Pb}][y_{Pb}]$ is equal to 1, the derivation process for luma motion vectors for merge mode as specified in clause 8.5.3.2.2 is invoked with the luma location (x_{Cb} , y_{Cb}), the luma location (x_{Pb} , y_{Pb}), the variables n_{CbS} , n_{PbW} , n_{PbH} and the partition index $partIdx$ as inputs, and the output being the luma motion vectors $mvL0$, $mvL1$, the reference indices $refIdxL0$, $refIdxL1$ and the prediction list utilization flags $predFlagL0$ and $predFlagL1$.
- Otherwise, for X being replaced by either 0 or 1 in the variables $predFlagLX$, $mvLX$ and $refIdxLX$, in $PRED_LX$, and in the syntax elements ref_idx_LX and $MvdLX$, the following ordered steps apply:

1. The variables $refIdxLX$ and $predFlagLX$ are derived as follows:

- If $inter_pred_idc[x_{Pb}][y_{Pb}]$ is equal to $PRED_LX$ or $PRED_BI$,

$$refIdxLX = ref_idx_LX[x_{Pb}][y_{Pb}] \quad (8-88)$$

$$predFlagLX = 1 \quad (8-89)$$

- Otherwise, the variables $refIdxLX$ and $predFlagLX$ are specified by:

$$\text{refIdxLX} = -1 \quad (8-90)$$

$$\text{predFlagLX} = 0 \quad (8-91)$$

2. The variable `mvdLX` is derived as follows:

$$\text{mvdLX}[0] = \text{MvdLX}[\text{xPb}][\text{yPb}][0] \quad (8-92)$$

$$\text{mvdLX}[1] = \text{MvdLX}[\text{xPb}][\text{yPb}][1] \quad (8-93)$$

3. When `predFlagLX` is equal to 1, the derivation process for luma motion vector prediction in clause 8.5.3.2.6 is invoked with the luma coding block location (`xCb`, `yCb`), the coding block size `nCbS`, the luma prediction block location (`xPb`, `yPb`), the variables `nPbW`, `nPbH`, `refIdxLX` and the partition index `partIdx` as inputs, and the output being `mvpLX`.
4. When `predFlagLX` is equal to 1 and the picture with index `refIdx` from reference picture list `LX` of the slice is not the current picture, and `use_integer_mv_flag` is equal to 0, the luma motion vector `mvLX` is derived as follows:

$$\text{uLX}[0] = (\text{mvpLX}[0] + \text{mvdLX}[0] + 2^{16}) \% 2^{16} \quad (8-94)$$

$$\text{mvLX}[0] = (\text{uLX}[0] \geq 2^{15}) ? (\text{uLX}[0] - 2^{16}) : \text{uLX}[0] \quad (8-95)$$

$$\text{uLX}[1] = (\text{mvpLX}[1] + \text{mvdLX}[1] + 2^{16}) \% 2^{16} \quad (8-96)$$

$$\text{mvLX}[1] = (\text{uLX}[1] \geq 2^{15}) ? (\text{uLX}[1] - 2^{16}) : \text{uLX}[1] \quad (8-97)$$

NOTE 1– The resulting values of `mvLX[0]` and `mvLX[1]` as specified above will always be in the range of -2^{15} to $2^{15} - 1$, inclusive.

5. When `predFlagLX` is equal to 1 and the reference picture is the current picture, or when `predFlagLX` is equal to 1 and `use_integer_mv_flag` is equal to 1, the luma motion vector `mvLX` is derived as follows:

$$\text{uLX}[0] = (((\text{mvpLX}[0] \gg 2) + \text{mvdLX}[0]) \ll 2) + 2^{16}) \% 2^{16} \quad (8-98)$$

$$\text{mvLX}[0] = (\text{uLX}[0] \geq 2^{15}) ? (\text{uLX}[0] - 2^{16}) : \text{uLX}[0] \quad (8-99)$$

$$\text{uLX}[1] = (((\text{mvpLX}[1] \gg 2) + \text{mvdLX}[1]) \ll 2) + 2^{16}) \% 2^{16} \quad (8-100)$$

$$\text{mvLX}[1] = (\text{uLX}[1] \geq 2^{15}) ? (\text{uLX}[1] - 2^{16}) : \text{uLX}[1] \quad (8-101)$$

NOTE 2 – The resulting values of `mvLX[0]` and `mvLX[1]` as specified above will always be in the range of -2^{15} to $2^{15} - 1$, inclusive.

- The variable `noIntegerMvFlag` is derived as follows:

$$\text{noIntegerMvFlag} = !((\text{mvL0} \& 0x3 == 0) \mid (\text{mvL1} \& 0x3 == 0)) \quad (8-102)$$

- The variable `identicalMvs` is derived as follows:

$$\text{identicalMvs} = (\text{mvL0} == \text{mvL1}) \&\& (\text{DiffPicOrderCnt}(\text{RefPicList0}[\text{refIdxL0}], \text{RefPicList1}[\text{refIdxL1}]) == 0) \quad (8-103)$$

When all of the following conditions are true, `refIdxL1` is set equal to -1 and `predFlagL1` is set equal to 0.

- `predFlagL0` is equal to 1.
- `predFlagL1` is equal to 1.
- `nPbSw` is equal to 8.
- `nPbSh` is equal to 8.
- `TwoVersionsOfCurrDecPicFlag` is equal to 1.

- noIntegerMvFlag is equal to 1.
- identicalMvs is equal to 0.

When ChromaArrayType is not equal to 0 and predFlagLX, with X being 0 or 1, is equal to 1, the derivation process for chroma motion vectors in clause 8.5.3.2.10 is invoked with mvLX as input, and the output being mvCLX.

The variables offsetX and offsetY are derived as follows:

$$\text{offsetX} = (\text{ChromaArrayType} == 0) ? 0 : (\text{mvCLX}[0] \& 0x7 ? 2 : 0) \quad (8-104)$$

$$\text{offsetY} = (\text{ChromaArrayType} == 0) ? 0 : (\text{mvCLX}[1] \& 0x7 ? 2 : 0) \quad (8-105)$$

It is a requirement of bitstream conformance that when the reference picture is the current picture, the luma motion vector mvLX shall obey the following constraints:

- When the derivation process for z-scan order block availability as specified in clause 6.4.1 is invoked with (xCurr, yCurr) set equal to (xCb, yCb) and the neighbouring luma location (xNbY, yNbY) set equal to (xPb + (mvLX[0] >> 2) – offsetX, yPb + (mvLX[1] >> 2) – offsetY) as inputs, the output shall be equal to TRUE.
- When the derivation process for z-scan order block availability as specified in clause 6.4.1 is invoked with (xCurr, yCurr) set equal to (xCb, yCb) and the neighbouring luma location (xNbY, yNbY) set equal to (xPb + (mvLX[0] >> 2) + nPbW – 1 + offsetX, yPb + (mvLX[1] >> 2) + nPbH – 1 + offsetY) as inputs, the output shall be equal to TRUE.
- One or both of the following conditions shall be true:
 - The value of (mvLX[0] >> 2) + nPbW + xB1 + offsetX is less than or equal to 0.
 - The value of (mvLX[1] >> 2) + nPbH + yB1 + offsetY is less than or equal to 0.
- The following condition shall be true:

$$(\text{xPb} + (\text{mvLX}[0] \gg 2) + \text{nPbSw} - 1 + \text{offsetX}) / \text{CtbSizeY} - \text{xCb} / \text{CtbSizeY} \leq \text{yCb} / \text{CtbSizeY} - (\text{yPb} + (\text{mvLX}[1] \gg 2) + \text{nPbSh} - 1 + \text{offsetY}) / \text{CtbSizeY} \quad (8-106)$$

8.5.3.2.2 Derivation process for luma motion vectors for merge mode

This process is only invoked when merge_flag[xPb][yPb] is equal to 1, where (xPb, yPb) specify the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture.

Inputs to this process are:

- a luma location (xCb, yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a luma location (xPb, yPb) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
- a variable nCbS specifying the size of the current luma coding block,
- two variables nPbW and nPbH specifying the width and the height of the luma prediction block,
- a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

- the luma motion vectors mvL0 and mvL1,
- the reference indices refIdxL0 and refIdxL1,
- the prediction list utilization flags predFlagL0 and predFlagL1.

The location (xOrigP, yOrigP) and the variables nOrigPbW and nOrigPbH are derived to store the values of (xPb, yPb), nPbW and nPbH as follows:

$$(\text{xOrigP}, \text{yOrigP}) \text{ is set equal to } (\text{xPb}, \text{yPb}) \quad (8-107)$$

$$\text{nOrigPbW} = \text{nPbW} \quad (8-108)$$

$$nOrigPbH = nPbH \quad (8-109)$$

When Log2ParMrgLevel is greater than 2 and nCbS is equal to 8, (xPb, yPb), nPbW, nPbH and partIdx are modified as follows:

$$(xPb, yPb) = (xCb, yCb) \quad (8-110)$$

$$nPbW = nCbS \quad (8-111)$$

$$nPbH = nCbS \quad (8-112)$$

$$partIdx = 0 \quad (8-113)$$

NOTE – When Log2ParMrgLevel is greater than 2 and nCbS is equal to 8, all the prediction units of the current coding unit share a single merge candidate list, which is identical to the merge candidate list of the 2Nx2N prediction unit.

The motion vectors mvL0 and mvL1, the reference indices refIdxL0 and refIdxL1 and the prediction utilization flags predFlagL0 and predFlagL1 are derived by the following ordered steps:

1. The derivation process for merging candidates from neighbouring prediction unit partitions in clause 8.5.3.2.3 is invoked with the luma coding block location (xCb, yCb), the coding block size nCbS, the luma prediction block location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH and the partition index partIdx as inputs, and the output being the availability flags availableFlagA₀, availableFlagA₁, availableFlagB₀, availableFlagB₁ and availableFlagB₂, the reference indices refIdxLXA₀, refIdxLXA₁, refIdxLXB₀, refIdxLXB₁ and refIdxLXB₂, the prediction list utilization flags predFlagLXA₀, predFlagLXA₁, predFlagLXB₀, predFlagLXB₁ and predFlagLXB₂, and the motion vectors mvLXA₀, mvLXA₁, mvLXB₀, mvLXB₁ and mvLXB₂, with X being 0 or 1.
2. The reference indices for the temporal merging candidate, refIdxLXCol, with X being 0 or 1, are set equal to 0.
3. The derivation process for temporal luma motion vector prediction in clause 8.5.3.2.8 is invoked with the luma location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH and the variable refIdxL0Col as inputs, and the output being the availability flag availableFlagL0Col and the temporal motion vector mvL0Col. The variables availableFlagCol, predFlagL0Col and predFlagL1Col are derived as follows:

$$availableFlagCol = availableFlagL0Col \quad (8-114)$$

$$predFlagL0Col = availableFlagL0Col \quad (8-115)$$

$$predFlagL1Col = 0 \quad (8-116)$$

4. When slice_type is equal to B, the derivation process for temporal luma motion vector prediction in clause 8.5.3.2.8 is invoked with the luma location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH and the variable refIdxL1Col as inputs, and the output being the availability flag availableFlagL1Col and the temporal motion vector mvL1Col. The variables availableFlagCol and predFlagL1Col are derived as follows:

$$availableFlagCol = availableFlagL0Col \ || \ availableFlagL1Col \quad (8-117)$$

$$predFlagL1Col = availableFlagL1Col \quad (8-118)$$

5. The merging candidate list, mergeCandList, is constructed as follows:

$$\begin{aligned} i &= 0 \\ \text{if}(availableFlagA_1) \\ &\quad mergeCandList[i++] = A_1 \\ \text{if}(availableFlagB_1) \\ &\quad mergeCandList[i++] = B_1 \\ \text{if}(availableFlagB_0) \\ &\quad mergeCandList[i++] = B_0 \\ \text{if}(availableFlagA_0) \\ &\quad mergeCandList[i++] = A_0 \end{aligned} \quad (8-119)$$

```

if( availableFlagB2 )
    mergeCandList[ i++ ] = B2
if( availableFlagCol )
    mergeCandList[ i++ ] = Col

```

6. The variable numCurrMergeCand and numOrigMergeCand are set equal to the number of merging candidates in the mergeCandList.
7. When slice_type is equal to B, the derivation process for combined bi-predictive merging candidates specified in clause 8.5.3.2.4 is invoked with mergeCandList, the reference indices refIdxL0N and refIdxL1N, the prediction list utilization flags predFlagL0N and predFlagL1N, the motion vectors mvL0N and mvL1N of every candidate N in mergeCandList, numCurrMergeCand and numOrigMergeCand as inputs, and the output is assigned to mergeCandList, numCurrMergeCand, the reference indices refIdxL0combCand_k and refIdxL1combCand_k, the prediction list utilization flags predFlagL0combCand_k and predFlagL1combCand_k and the motion vectors mvL0combCand_k and mvL1combCand_k of every new candidate combCand_k being added into mergeCandList. The number of candidates being added, numCombMergeCand, is set equal to (numCurrMergeCand – numOrigMergeCand). When numCombMergeCand is greater than 0, k ranges from 0 to numCombMergeCand – 1, inclusive.
8. The derivation process for zero motion vector merging candidates specified in clause 8.5.3.2.5 is invoked with the mergeCandList, the reference indices refIdxL0N and refIdxL1N, the prediction list utilization flags predFlagL0N and predFlagL1N, the motion vectors mvL0N and mvL1N of every candidate N in mergeCandList and numCurrMergeCand as inputs, and the output is assigned to mergeCandList, numCurrMergeCand, the reference indices refIdxL0zeroCand_m and refIdxL1zeroCand_m, the prediction list utilization flags predFlagL0zeroCand_m and predFlagL1zeroCand_m and the motion vectors mvL0zeroCand_m and mvL1zeroCand_m of every new candidate zeroCand_m being added into mergeCandList. The number of candidates being added, numZeroMergeCand, is set equal to (numCurrMergeCand – numOrigMergeCand – numCombMergeCand). When numZeroMergeCand is greater than 0, m ranges from 0 to numZeroMergeCand – 1, inclusive.
9. The following assignments are made with N being the candidate at position merge_idx[xOrigP][yOrigP] in the merging candidate list mergeCandList (N = mergeCandList[merge_idx[xOrigP][yOrigP]]) and X being replaced by 0 or 1:

$$\text{refIdxLX} = \text{refIdxLXN} \quad (8-120)$$

$$\text{predFlagLX} = \text{predFlagLXN} \quad (8-121)$$

- If use_integer_mv_flag is equal to 0 and the reference picture is not the current picture, the following applies:

$$\text{mvLX}[0] = \text{mvLXN}[0] \quad (8-122)$$

$$\text{mvLX}[1] = \text{mvLXN}[1] \quad (8-123)$$

- Otherwise (use_integer_mv_flag is equal to 1 or the reference picture is the current picture), the following applies:

$$\text{mvLX}[0] = (\text{mvLXN}[0] \gg 2) \ll 2 \quad (8-124)$$

$$\text{mvLX}[1] = (\text{mvLXN}[1] \gg 2) \ll 2 \quad (8-125)$$

10. When predFlagL0 is equal to 1 and predFlagL1 is equal to 1 and (nOrigPbW + nOrigPbH) is equal to 12, the following applies:

$$\text{refIdxL1} = -1 \quad (8-126)$$

$$\text{predFlagL1} = 0 \quad (8-127)$$

8.5.3.2.3 Derivation process for spatial merging candidates

Inputs to this process are:

- a luma location (xCb, yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a variable nCbS specifying the size of the current luma coding block,

- a luma location (x_{Pb} , y_{Pb}) specifying the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
- two variables n_{PbW} and n_{PbH} specifying the width and the height of the luma prediction block,
- a variable $partIdx$ specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are as follows, with X being 0 or 1:

- the availability flags $availableFlagA_0$, $availableFlagA_1$, $availableFlagB_0$, $availableFlagB_1$ and $availableFlagB_2$ of the neighbouring prediction units,
- the reference indices $refIdxLXA_0$, $refIdxLXA_1$, $refIdxLXB_0$, $refIdxLXB_1$ and $refIdxLXB_2$ of the neighbouring prediction units,
- the prediction list utilization flags $predFlagLXA_0$, $predFlagLXA_1$, $predFlagLXB_0$, $predFlagLXB_1$ and $predFlagLXB_2$ of the neighbouring prediction units,
- the motion vectors $mvLXA_0$, $mvLXA_1$, $mvLXB_0$, $mvLXB_1$ and $mvLXB_2$ of the neighbouring prediction units.

For the derivation of $availableFlagA_1$, $refIdxLXA_1$, $predFlagLXA_1$ and $mvLXA_1$ the following applies:

- The luma location (x_{NbA_1} , y_{NbA_1}) inside the neighbouring luma coding block is set equal to ($x_{Pb} - 1$, $y_{Pb} + n_{PbH} - 1$).
- The availability derivation process for a prediction block as specified in clause 6.4.2 is invoked with the luma location (x_{Cb} , y_{Cb}), the current luma coding block size n_{CbS} , the luma prediction block location (x_{Pb} , y_{Pb}), the luma prediction block width n_{PbW} , the luma prediction block height n_{PbH} , the luma location (x_{NbA_1} , y_{NbA_1}) and the partition index $partIdx$ as inputs, and the output is assigned to the prediction block availability flag $availableA_1$.
- When one or more of the following conditions are true, $availableA_1$ is set equal to FALSE:
 - $x_{Pb} \gg \text{Log2ParMrgLevel}$ is equal to $x_{NbA_1} \gg \text{Log2ParMrgLevel}$ and $y_{Pb} \gg \text{Log2ParMrgLevel}$ is equal to $y_{NbA_1} \gg \text{Log2ParMrgLevel}$.
 - $PartMode$ of the current prediction unit is equal to $PART_Nx2N$, $PART_nLx2N$ or $PART_nRx2N$ and $partIdx$ is equal to 1.
- The variables $availableFlagA_1$, $refIdxLXA_1$, $predFlagLXA_1$ and $mvLXA_1$ are derived as follows:
 - If $availableA_1$ is equal to FALSE, $availableFlagA_1$ is set equal to 0, both components of $mvLXA_1$ are set equal to 0, $refIdxLXA_1$ is set equal to -1 and $predFlagLXA_1$ is set equal to 0, with X being 0 or 1.
 - Otherwise, $availableFlagA_1$ is set equal to 1 and the following assignments are made:

$$mvLXA_1 = MvLX[x_{NbA_1}][y_{NbA_1}] \quad (8-128)$$

$$refIdxLXA_1 = RefIdxLX[x_{NbA_1}][y_{NbA_1}] \quad (8-129)$$

$$predFlagLXA_1 = PredFlagLX[x_{NbA_1}][y_{NbA_1}] \quad (8-130)$$

For the derivation of $availableFlagB_1$, $refIdxLXB_1$, $predFlagLXB_1$ and $mvLXB_1$ the following applies:

- The luma location (x_{NbB_1} , y_{NbB_1}) inside the neighbouring luma coding block is set equal to ($x_{Pb} + n_{PbW} - 1$, $y_{Pb} - 1$).
- The availability derivation process for a prediction block as specified in clause 6.4.2 is invoked with the luma location (x_{Cb} , y_{Cb}), the current luma coding block size n_{CbS} , the luma prediction block location (x_{Pb} , y_{Pb}), the luma prediction block width n_{PbW} , the luma prediction block height n_{PbH} , the luma location (x_{NbB_1} , y_{NbB_1}) and the partition index $partIdx$ as inputs, and the output is assigned to the prediction block availability flag $availableB_1$.
- When one or more of the following conditions are true, $availableB_1$ is set equal to FALSE:
 - $x_{Pb} \gg \text{Log2ParMrgLevel}$ is equal to $x_{NbB_1} \gg \text{Log2ParMrgLevel}$ and $y_{Pb} \gg \text{Log2ParMrgLevel}$ is equal to $y_{NbB_1} \gg \text{Log2ParMrgLevel}$.
 - $PartMode$ of the current prediction unit is equal to $PART_2NxN$, $PART_2NxN_U$ or $PART_2NxN_D$ and $partIdx$ is equal to 1.
- The variables $availableFlagB_1$, $refIdxLXB_1$, $predFlagLXB_1$ and $mvLXB_1$ are derived as follows:

- If one or more of the following conditions are true, availableFlagB₁ is set equal to 0, both components of mvLXB₁ are set equal to 0, refIdxLXB₁ is set equal to –1 and predFlagLXB₁ is set equal to 0, with X being 0 or 1:
 - availableB₁ is equal to FALSE.
 - availableA₁ is equal to TRUE and the prediction units covering the luma locations (xNbA₁, yNbA₁) and (xNbB₁, yNbB₁) have the same motion vectors and the same reference indices.
- Otherwise, availableFlagB₁ is set equal to 1 and the following assignments are made:

$$mvLXB_1 = MvLX[xNbB_1][yNbB_1] \quad (8-131)$$

$$refIdxLXB_1 = RefIdxLX[xNbB_1][yNbB_1] \quad (8-132)$$

$$predFlagLXB_1 = PredFlagLX[xNbB_1][yNbB_1] \quad (8-133)$$

For the derivation of availableFlagB₀, refIdxLXB₀, predFlagLXB₀ and mvLXB₀ the following applies:

- The luma location (xNbB₀, yNbB₀) inside the neighbouring luma coding block is set equal to (xPb + nPbW, yPb – 1).
- The availability derivation process for a prediction block as specified in clause 6.4.2 is invoked with the luma location (xCb, yCb), the current luma coding block size nCbS, the luma prediction block location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH, the luma location (xNbB₀, yNbB₀) and the partition index partIdx as inputs, and the output is assigned to the prediction block availability flag availableB₀.
- When xPb >> Log2ParMrgLevel is equal to xNbB₀ >> Log2ParMrgLevel and yPb >> Log2ParMrgLevel is equal to yNbB₀ >> Log2ParMrgLevel, availableB₀ is set equal to FALSE.
- The variables availableFlagB₀, refIdxLXB₀, predFlagLXB₀ and mvLXB₀ are derived as follows:
 - If one or more of the following conditions are true, availableFlagB₀ is set equal to 0, both components of mvLXB₀ are set equal to 0, refIdxLXB₀ is set equal to –1 and predFlagLXB₀ is set equal to 0, with X being 0 or 1:
 - availableB₀ is equal to FALSE.
 - availableB₁ is equal to TRUE and the prediction units covering the luma locations (xNbB₁, yNbB₁) and (xNbB₀, yNbB₀) have the same motion vectors and the same reference indices.
 - Otherwise, availableFlagB₀ is set equal to 1 and the following assignments are made:

$$mvLXB_0 = MvLX[xNbB_0][yNbB_0] \quad (8-134)$$

$$refIdxLXB_0 = RefIdxLX[xNbB_0][yNbB_0] \quad (8-135)$$

$$predFlagLXB_0 = PredFlagLX[xNbB_0][yNbB_0] \quad (8-136)$$

For the derivation of availableFlagA₀, refIdxLXA₀, predFlagLXA₀ and mvLXA₀ the following applies:

- The luma location (xNbA₀, yNbA₀) inside the neighbouring luma coding block is set equal to (xPb – 1, yPb + nPbH).
- The availability derivation process for a prediction block as specified in clause 6.4.2 is invoked with the luma location (xCb, yCb), the current luma coding block size nCbS, the luma prediction block location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH, the luma location (xNbA₀, yNbA₀) and the partition index partIdx as inputs, and the output is assigned to the prediction block availability flag availableA₀.
- When xPb >> Log2ParMrgLevel is equal to xNbA₀ >> Log2ParMrgLevel and yPb >> Log2ParMrgLevel is equal to yA₀ >> Log2ParMrgLevel, availableA₀ is set equal to FALSE.
- The variables availableFlagA₀, refIdxLXA₀, predFlagLXA₀ and mvLXA₀ are derived as follows:
 - If one or more of the following conditions are true, availableFlagA₀ is set equal to 0, both components of mvLXA₀ are set equal to 0, refIdxLXA₀ is set equal to –1 and predFlagLXA₀ is set equal to 0, with X being 0 or 1:
 - availableA₀ is equal to FALSE.

- availableA₁ is equal to TRUE and the prediction units covering the luma locations (xNbA₁, yNbA₁) and (xNbA₀, yNbA₀) have the same motion vectors and the same reference indices.
- Otherwise, availableFlagA₀ is set equal to 1 and the following assignments are made:

$$mvLXA_0 = MvLX[xNbA_0][yNbA_0] \quad (8-137)$$

$$refIdxLXA_0 = RefIdxLX[xNbA_0][yNbA_0] \quad (8-138)$$

$$predFlagLXA_0 = PredFlagLX[xNbA_0][yNbA_0] \quad (8-139)$$

For the derivation of availableFlagB₂, refIdxLXB₂, predFlagLXB₂ and mvLXB₂ the following applies:

- The luma location (xNbB₂, yNbB₂) inside the neighbouring luma coding block is set equal to (xPb – 1, yPb – 1).
- The availability derivation process for a prediction block as specified in clause 6.4.2 is invoked with the luma location (xCb, yCb), the current luma coding block size nCbS, the luma prediction block location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH, the luma location (xNbB₂, yNbB₂) and the partition index partIdx as inputs, and the output is assigned to the prediction block availability flag availableB₂.
- When xPb >> Log2ParMrgLevel is equal to xNbB₂ >> Log2ParMrgLevel and yPb >> Log2ParMrgLevel is equal to yNbB₂ >> Log2ParMrgLevel, availableB₂ is set equal to FALSE.
- The variables availableFlagB₂, refIdxLXB₂, predFlagLXB₂ and mvLXB₂ are derived as follows:
 - If one or more of the following conditions are true, availableFlagB₂ is set equal to 0, both components of mvLXB₂ are set equal to 0, refIdxLXB₂ is set equal to –1 and predFlagLXB₂ is set equal to 0, with X being 0 or 1:
 - availableB₂ is equal to FALSE.
 - availableA₁ is equal to TRUE and prediction units covering the luma locations (xNbA₁, yNbA₁) and (xNbB₂, yNbB₂) have the same motion vectors and the same reference indices.
 - availableB₁ is equal to TRUE and the prediction units covering the luma locations (xNbB₁, yNbB₁) and (xNbB₂, yNbB₂) have the same motion vectors and the same reference indices.
 - availableFlagA₀ + availableFlagA₁ + availableFlagB₀ + availableFlagB₁ is equal to 4.
 - Otherwise, availableFlagB₂ is set equal to 1 and the following assignments are made:

$$mvLXB_2 = MvLX[xNbB_2][yNbB_2] \quad (8-140)$$

$$refIdxLXB_2 = RefIdxLX[xNbB_2][yNbB_2] \quad (8-141)$$

$$predFlagLXB_2 = PredFlagLX[xNbB_2][yNbB_2] \quad (8-142)$$

8.5.3.2.4 Derivation process for combined bi-predictive merging candidates

Inputs to this process are:

- a merging candidate list mergeCandList,
- the reference indices refIdxL0N and refIdxL1N of every candidate N in mergeCandList,
- the prediction list utilization flags predFlagL0N and predFlagL1N of every candidate N in mergeCandList,
- the motion vectors mvL0N and mvL1N of every candidate N in mergeCandList,
- the number of elements numCurrMergeCand within mergeCandList,
- the number of elements numOrigMergeCand within the mergeCandList after the spatial and temporal merge candidate derivation process.

Outputs of this process are:

- the merging candidate list mergeCandList,
- the number of elements numCurrMergeCand within mergeCandList,
- the reference indices refIdxL0combCand_k and refIdxL1combCand_k of every new candidate combCand_k added into mergeCandList during the invocation of this process,

- the prediction list utilization flags $\text{predFlagL0combCand}_k$ and $\text{predFlagL1combCand}_k$ of every new candidate combCand_k added into mergeCandList during the invocation of this process,
- the motion vectors mvL0combCand_k and mvL1combCand_k of every new candidate combCand_k added into mergeCandList during the invocation of this process.

When numOrigMergeCand is greater than 1 and less than MaxNumMergeCand , the variable numInputMergeCand is set equal to numCurrMergeCand , the variable combIdx is set equal to 0, the variable combStop is set equal to FALSE and the following ordered steps are repeated until combStop is equal to TRUE:

1. The variables l0CandIdx and l1CandIdx are derived using combIdx as specified in Table 8-7.
2. The following assignments are made, with l0Cand being the candidate at position l0CandIdx and l1Cand being the candidate at position l1CandIdx in the merging candidate list mergeCandList :
 - $\text{l0Cand} = \text{mergeCandList}[\text{l0CandIdx}]$
 - $\text{l1Cand} = \text{mergeCandList}[\text{l1CandIdx}]$
3. When all of the following conditions are true:
 - $\text{predFlagL0l0Cand} == 1$
 - $\text{predFlagL1l1Cand} == 1$
 - $(\text{DiffPicOrderCnt}(\text{RefPicList0}[\text{refIdxL0l0Cand}], \text{RefPicList1}[\text{refIdxL1l1Cand}]) \neq 0) \quad || \quad (\text{mvL0l0Cand} \neq \text{mvL1l1Cand})$

the candidate combCand_k with k equal to $(\text{numCurrMergeCand} - \text{numInputMergeCand})$ is added at the end of mergeCandList , i.e., $\text{mergeCandList}[\text{numCurrMergeCand}]$ is set equal to combCand_k , and the reference indices, the prediction list utilization flags and the motion vectors of combCand_k are derived as follows and numCurrMergeCand is incremented by 1:

$$\text{refIdxL0combCand}_k = \text{refIdxL0l0Cand} \quad (8-143)$$

$$\text{refIdxL1combCand}_k = \text{refIdxL1l1Cand} \quad (8-144)$$

$$\text{predFlagL0combCand}_k = 1 \quad (8-145)$$

$$\text{predFlagL1combCand}_k = 1 \quad (8-146)$$

$$\text{mvL0combCand}_k[0] = \text{mvL0l0Cand}[0] \quad (8-147)$$

$$\text{mvL0combCand}_k[1] = \text{mvL0l0Cand}[1] \quad (8-148)$$

$$\text{mvL1combCand}_k[0] = \text{mvL1l1Cand}[0] \quad (8-149)$$

$$\text{mvL1combCand}_k[1] = \text{mvL1l1Cand}[1] \quad (8-150)$$

$$\text{numCurrMergeCand} = \text{numCurrMergeCand} + 1 \quad (8-151)$$

4. The variable combIdx is incremented by 1.
5. When combIdx is equal to $(\text{numOrigMergeCand} * (\text{numOrigMergeCand} - 1))$ or numCurrMergeCand is equal to MaxNumMergeCand , combStop is set equal to TRUE.

Table 8-7 – Specification of l0CandIdx and l1CandIdx

combIdx	0	1	2	3	4	5	6	7	8	9	10	11
l0CandIdx	0	1	0	2	1	2	0	3	1	3	2	3
l1CandIdx	1	0	2	0	2	1	3	0	3	1	3	2

8.5.3.2.5 Derivation process for zero motion vector merging candidates

Inputs to this process are:

- a merging candidate list mergeCandList,
- the reference indices refIdxL0N and refIdxL1N of every candidate N in mergeCandList,
- the prediction list utilization flags predFlagL0N and predFlagL1N of every candidate N in mergeCandList,
- the motion vectors mvL0N and mvL1N of every candidate N in mergeCandList,
- the number of elements numCurrMergeCand within mergeCandList.

Outputs of this process are:

- the merging candidate list mergeCandList,
- the number of elements numCurrMergeCand within mergeCandList,
- the reference indices refIdxL0zeroCand_m and refIdxL1zeroCand_m of every new candidate zeroCand_m added into mergeCandList during the invocation of this process,
- the prediction list utilization flags predFlagL0zeroCand_m and predFlagL1zeroCand_m of every new candidate zeroCand_m added into mergeCandList during the invocation of this process,
- the motion vectors mvL0zeroCand_m and mvL1zeroCand_m of every new candidate zeroCand_m added into mergeCandList during the invocation of this process.

The variable numRefIdx is derived as follows:

- If slice_type is equal to P, numRefIdx is set equal to num_ref_idx_l0_active_minus1 + 1.
- Otherwise (slice_type is equal to B), numRefIdx is set equal to Min(num_ref_idx_l0_active_minus1 + 1, num_ref_idx_l1_active_minus1 + 1).

When numCurrMergeCand is less than MaxNumMergeCand, the variable numInputMergeCand is set equal to numCurrMergeCand, the variable zeroIdx is set equal to 0 and the following ordered steps are repeated until numCurrMergeCand is equal to MaxNumMergeCand:

1. For the derivation of the reference indices, the prediction list utilization flags and the motion vectors of the zero motion vector merging candidate, the following applies:
 - If slice_type is equal to P, the candidate zeroCand_m with m equal to (numCurrMergeCand – numInputMergeCand) is added at the end of mergeCandList, i.e., mergeCandList[numCurrMergeCand] is set equal to zeroCand_m, and the reference indices, the prediction list utilization flags and the motion vectors of zeroCand_m are derived as follows and numCurrMergeCand is incremented by 1:

$$\text{refIdxL0zeroCand}_m = (\text{zeroIdx} < \text{numRefIdx}) ? \text{zeroIdx} : 0 \quad (8-152)$$

$$\text{refIdxL1zeroCand}_m = -1 \quad (8-153)$$

$$\text{predFlagL0zeroCand}_m = 1 \quad (8-154)$$

$$\text{predFlagL1zeroCand}_m = 0 \quad (8-155)$$

$$\text{mvL0zeroCand}_m[0] = 0 \quad (8-156)$$

$$\text{mvL0zeroCand}_m[1] = 0 \quad (8-157)$$

$$\text{mvL1zeroCand}_m[0] = 0 \quad (8-158)$$

$$\text{mvL1zeroCand}_m[1] = 0 \quad (8-159)$$

$$\text{numCurrMergeCand} = \text{numCurrMergeCand} + 1 \quad (8-160)$$

- Otherwise (slice_type is equal to B), the candidate zeroCand_m with m equal to (numCurrMergeCand – numInputMergeCand) is added at the end of mergeCandList, i.e.,

mergeCandList[numCurrMergeCand] is set equal to zeroCand_m, and the reference indices, the prediction list utilization flags and the motion vectors of zeroCand_m are derived as follows and numCurrMergeCand is incremented by 1:

$$\text{refIdxL0zeroCand}_m = (\text{zeroIdx} < \text{numRefIdx}) ? \text{zeroIdx} : 0 \quad (8-161)$$

$$\text{refIdxL1zeroCand}_m = (\text{zeroIdx} < \text{numRefIdx}) ? \text{zeroIdx} : 0 \quad (8-162)$$

$$\text{predFlagL0zeroCand}_m = 1 \quad (8-163)$$

$$\text{predFlagL1zeroCand}_m = 1 \quad (8-164)$$

$$\text{mvL0zeroCand}_m[0] = 0 \quad (8-165)$$

$$\text{mvL0zeroCand}_m[1] = 0 \quad (8-166)$$

$$\text{mvL1zeroCand}_m[0] = 0 \quad (8-167)$$

$$\text{mvL1zeroCand}_m[1] = 0 \quad (8-168)$$

$$\text{numCurrMergeCand} = \text{numCurrMergeCand} + 1 \quad (8-169)$$

2. The variable zeroIdx is incremented by 1.

8.5.3.2.6 Derivation process for luma motion vector prediction

Inputs to this process are:

- a luma location (xCb, yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a variable nCbS specifying the size of the current luma coding block,
- a luma location (xPb, yPb) specifying the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
- two variables nPbW and nPbH specifying the width and the height of the luma prediction block,
- the reference index of the current prediction unit partition refIdxLX, with X being 0 or 1,
- a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Output of this process is the prediction mvpLX of the motion vector mvLX, with X being 0 or 1.

The motion vector predictor mvpLX is derived in the following ordered steps:

1. The derivation process for motion vector predictor candidates from neighbouring prediction unit partitions in clause 8.5.3.2.7 is invoked with the luma coding block location (xCb, yCb), the coding block size nCbS, the luma prediction block location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH, refIdxLX, with X being 0 or 1 and the partition index partIdx as inputs, and the availability flags availableFlagLXN and the motion vectors mvLXN, with N being replaced by A or B, as output.
2. If both availableFlagLXA and availableFlagLXB are equal to 1 and mvLXA is not equal to mvLXB, availableFlagLXCol is set equal to 0. Otherwise, the derivation process for temporal luma motion vector prediction in clause 8.5.3.2.8 is invoked with luma prediction block location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH and refIdxLX, with X being 0 or 1 as inputs, and with the output being the availability flag availableFlagLXCol and the temporal motion vector predictor mvLXCol.
3. The motion vector predictor candidate list, mvpListLX, is constructed as follows:

```

i = 0
if( availableFlagLXA ) {
    mvpListLX[ i++ ] = mvLXA
    if( availableFlagLXB && ( mvLXA != mvLXB ) )
        mvpListLX[ i++ ] = mvLXB
} else if( availableFlagLXB )
    mvpListLX[ i++ ] = mvLXB

```

(8-170)

```

if( i < 2 && availableFlagLXCol )
   .mvpListLX[ i++ ] = mvLXCol
while( i < 2 ) {
   .mvpListLX[ i ][ 0 ] = 0
   .mvpListLX[ i ][ 1 ] = 0
    i++
}

```

4. The motion vector of $\text{mvListLX}[\text{mv_lX_flag}[\text{xPb}][\text{yPb}]]$ is assigned to mvLX .

8.5.3.2.7 Derivation process for motion vector predictor candidates

Inputs to this process are:

- a luma location (xCb, yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a variable nCbS specifying the size of the current luma coding block,
- a luma location (xPb, yPb) specifying the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
- two variables nPbW and nPbH specifying the width and the height of the luma prediction block,
- the reference index of the current prediction unit partition refIdxLX , with X being 0 or 1,
- a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are (with N being replaced by A or B):

- the motion vectors mvLXN of the neighbouring prediction units,
- the availability flags availableFlagLXN of the neighbouring prediction units.

Figure 8-3 provides an overview of spatial motion vector neighbours.

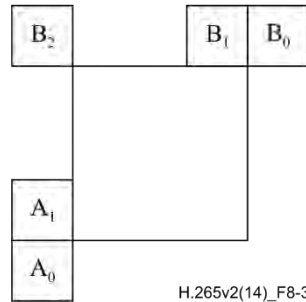


Figure 8-3 – Spatial motion vector neighbours (informative)

The variable currPb specifies the current luma prediction block at luma location (xPb, yPb) and the variable currPic specifies the current picture.

The variable isScaledFlagLX , with X being 0 or 1, is set equal to 0.

The motion vector mvLXA and the availability flag availableFlagLXA are derived in the following ordered steps:

1. The sample location ($\text{xNbA}_0, \text{yNbA}_0$) is set equal to ($\text{xPb} - 1, \text{yPb} + \text{nPbH}$) and the sample location ($\text{xNbA}_1, \text{yNbA}_1$) is set equal to ($\text{xNbA}_0, \text{yNbA}_0 - 1$).
2. The availability flag availableFlagLXA is set equal to 0 and both components of mvLXA are set equal to 0.
3. The availability derivation process for a prediction block as specified in clause 6.4.2 is invoked with the luma location (xCb, yCb), the current luma coding block size nCbS , the luma prediction block location (xPb, yPb), the luma prediction block width nPbW , the luma prediction block height nPbH , the luma location (xNbY, yNbY) set equal to ($\text{xNbA}_0, \text{yNbA}_0$) and the partition index partIdx as inputs, and the output is assigned to the prediction block availability flag availableA_0 .
4. The availability derivation process for a prediction block as specified in clause 6.4.2 is invoked with the luma location (xCb, yCb), the current luma coding block size nCbS , the luma prediction block location (xPb, yPb),

the luma prediction block width nPbW, the luma prediction block height nPbH, the luma location (xNbY, yNbY) set equal to (xNbA₁, yNbA₁) and the partition index partIdx as inputs, and the output is assigned to the prediction block availability flag availableA₁.

5. When availableA₀ or availableA₁ is equal to TRUE, the variable isScaledFlagLX is set equal to 1.

6. The following applies for (xNbA_k, yNbA_k) from (xNbA₀, yNbA₀) to (xNbA₁, yNbA₁):

– When availableA_k is equal to TRUE and availableFlagLXA is equal to 0, the following applies:

– If PredFlagLX[xNbA_k][yNbA_k] is equal to 1 and DiffPicOrderCnt(RefPicListX[RefIdxLX[xNbA_k][yNbA_k]], RefPicListX[refIdxLX]) is equal to 0, availableFlagLXA is set equal to 1 and the following applies:

$$mvLXA = MvLX[xNbA_k][yNbA_k] \quad (8-171)$$

– Otherwise, when PredFlagLY[xNbA_k][yNbA_k] (with Y = !X) is equal to 1 and DiffPicOrderCnt(RefPicListY[RefIdxLY[xNbA_k][yNbA_k]], RefPicListX[refIdxLX]) is equal to 0, availableFlagLXA is set equal to 1 and the following applies:

$$mvLXA = MvLY[xNbA_k][yNbA_k] \quad (8-172)$$

7. When availableFlagLXA is equal to 0, the following applies for (xNbA_k, yNbA_k) from (xNbA₀, yNbA₀) to (xNbA₁, yNbA₁) or until availableFlagLXA is equal to 1:

– When availableA_k is equal to TRUE and availableFlagLXA is equal to 0, the following applies:

– If PredFlagLX[xNbA_k][yNbA_k] is equal to 1 and LongTermRefPic(currPic, currPb, refIdxLX, RefPicListX) is equal to LongTermRefPic(currPic, currPb, RefIdxLX[xNbA_k][yNbA_k], RefPicListX), availableFlagLXA is set equal to 1 and the following assignments are made:

$$mvLXA = MvLX[xNbA_k][yNbA_k] \quad (8-173)$$

$$refIdxA = RefIdxLX[xNbA_k][yNbA_k] \quad (8-174)$$

$$refPicListA = RefPicListX \quad (8-175)$$

– Otherwise, when PredFlagLY[xNbA_k][yNbA_k] (with Y = !X) is equal to 1 and LongTermRefPic(currPic, currPb, refIdxLX, RefPicListX) is equal to LongTermRefPic(currPic, currPb, RefIdxLY[xNbA_k][yNbA_k], RefPicListY), availableFlagLXA is set equal to 1 and the following assignments are made:

$$mvLXA = MvLY[xNbA_k][yNbA_k] \quad (8-176)$$

$$refIdxA = RefIdxLY[xNbA_k][yNbA_k] \quad (8-177)$$

$$refPicListA = RefPicListY \quad (8-178)$$

– When availableFlagLXA is equal to 1, DiffPicOrderCnt(refPicListA[refIdxA], RefPicListX[refIdxLX]) is not equal to 0, and both refPicListA[refIdxA] and RefPicListX[refIdxLX] are short-term reference pictures, mvLXA is derived as follows:

$$tx = (16384 + (Abs(td) >> 1)) / td \quad (8-179)$$

$$distScaleFactor = Clip3(-4096, 4095, (tb * tx + 32) >> 6) \quad (8-180)$$

$$mvLXA = Clip3(-32768, 32767, Sign(distScaleFactor * mvLXA) * ((Abs(distScaleFactor * mvLXA) + 127) >> 8)) \quad (8-181)$$

where td and tb are derived as follows:

$$td = Clip3(-128, 127, DiffPicOrderCnt(currPic, refPicListA[refIdxA])) \quad (8-182)$$

$$tb = Clip3(-128, 127, DiffPicOrderCnt(currPic, RefPicListX[refIdxLX])) \quad (8-183)$$

The motion vector mvLXB and the availability flag availableFlagLXB are derived in the following ordered steps:

1. The sample locations (xNbB₀, yNbB₀), (xNbB₁, yNbB₁) and (xNbB₂, yNbB₂) are set equal to (xPb + nPbW, yPb - 1), (xPb + nPbW - 1, yPb - 1) and (xPb - 1, yPb - 1), respectively.
2. The availability flag availableFlagLXB is set equal to 0 and the both components of mvLXB are set equal to 0.
3. The following applies for (xNbB_k, yNbB_k) from (xNbB₀, yNbB₀) to (xNbB₂, yNbB₂):
 - The availability derivation process for a prediction block as specified in clause 6.4.2 is invoked with the luma location (xCb, yCb), the current luma coding block size nCbS, the luma prediction block location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH, the luma location (xNbY, yNbY) set equal to (xNbB_k, yNbB_k) and the partition index partIdx as inputs, and the output is assigned to the prediction block availability flag availableB_k.
 - When availableB_k is equal to TRUE and availableFlagLXB is equal to 0, the following applies:
 - If PredFlagLX[xNbB_k][yNbB_k] is equal to 1, and DiffPicOrderCnt(RefPicListX[RefIdxLX[xNbB_k][yNbB_k]], RefPicListX[refIdxLX]) is equal to 0, availableFlagLXB is set equal to 1 and the following assignment are made:

$$mvLXB = MvLX[xNbB_k][yNbB_k] \quad (8-184)$$

- Otherwise, when PredFlagLY[xNbB_k][yNbB_k] (with Y = !X) is equal to 1 and DiffPicOrderCnt(RefPicListY[RefIdxLY[xNbB_k][yNbB_k]], RefPicListX[refIdxLX]) is equal to 0, availableFlagLXB is set equal to 1 and the following assignment is made:

$$mvLXB = MvLY[xNbB_k][yNbB_k] \quad (8-185)$$

4. When isScaledFlagLX is equal to 0 and availableFlagLXB is equal to 1, availableFlagLXA is set equal to 1 and the following applies:

$$mvLXA = mvLXB \quad (8-186)$$

5. When isScaledFlagLX is equal to 0, availableFlagLXB is set equal to 0 and the following applies for (xNbB_k, yNbB_k) from (xNbB₀, yNbB₀) to (xNbB₂, yNbB₂) or until availableFlagLXB is equal to 1:

- The availability derivation process for a prediction block as specified in clause 6.4.2 is invoked with the luma location (xCb, yCb), the current luma coding block size nCbS, the luma location (xPb, yPb), the luma prediction block width nPbW, the luma prediction block height nPbH, the luma location (xNbY, yNbY) set equal to (xNbB_k, yNbB_k) and the partition index partIdx as inputs, and the output is assigned to the prediction block availability flag availableB_k.
- When availableB_k is equal to TRUE and availableFlagLXB is equal to 0, the following applies:
 - If PredFlagLX[xNbB_k][yNbB_k] is equal to 1 and LongTermRefPic(currPic, currPb, refIdxLX, RefPicListX) is equal to LongTermRefPic(currPic, currPb, RefIdxLX[xNbB_k][yNbB_k], RefPicListX), availableFlagLXB is set equal to 1 and the following assignments are made:

$$mvLXB = MvLX[xNbB_k][yNbB_k] \quad (8-187)$$

$$refIdxB = RefIdxLX[xNbB_k][yNbB_k] \quad (8-188)$$

$$refPicListB = RefPicListX \quad (8-189)$$

- Otherwise, when PredFlagLY[xNbB_k][yNbB_k] (with Y = !X) is equal to 1 and LongTermRefPic(currPic, currPb, refIdxLX, RefPicListX) is equal to LongTermRefPic(currPic, currPb, RefIdxLY[xNbB_k][yNbB_k], RefPicListY), availableFlagLXB is set equal to 1 and the following assignments are made:

$$mvLXB = MvLY[xNbB_k][yNbB_k] \quad (8-190)$$

$$refIdxB = RefIdxLY[xNbB_k][yNbB_k] \quad (8-191)$$

$$refPicListB = RefPicListY \quad (8-192)$$

- When availableFlagLXB is equal to 1, DiffPicOrderCnt(refPicListB[refIdxB], RefPicListX[refIdxLX]) is not equal to 0 and both refPicListB[refIdxB] and RefPicListX[refIdxLX] are short-term reference pictures, mvLXB is derived as follows:

$$tx = (16384 + (Abs(td) >> 1)) / td \quad (8-193)$$

$$distScaleFactor = Clip3(-4096, 4095, (tb * tx + 32) >> 6) \quad (8-194)$$

$$mvLXB = Clip3(-32768, 32767, Sign(distScaleFactor * mvLXB) * ((Abs(distScaleFactor * mvLXB) + 127) >> 8)) \quad (8-195)$$

where td and tb are derived as follows:

$$td = Clip3(-128, 127, DiffPicOrderCnt(currPic, refPicListB[refIdxB])) \quad (8-196)$$

$$tb = Clip3(-128, 127, DiffPicOrderCnt(currPic, RefPicListX[refIdxLX])) \quad (8-197)$$

8.5.3.2.8 Derivation process for temporal luma motion vector prediction

Inputs to this process are:

- a luma location (xPb, yPb) specifying the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
- two variables nPbW and nPbH specifying the width and the height of the luma prediction block,
- a reference index refIdxLX, with X being 0 or 1.

Outputs of this process are:

- the motion vector prediction mvLXCol,
- the availability flag availableFlagLXCol.

The variable currPb specifies the current luma prediction block at luma location (xPb, yPb).

The variables mvLXCol and availableFlagLXCol are derived as follows:

- If slice_temporal_mvp_enabled_flag is equal to 0, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.
- Otherwise (slice_temporal_mvp_enabled_flag is equal to 1), the following ordered steps apply:

1. The bottom right collocated motion vector is derived as follows:

$$xColBr = xPb + nPbW \quad (8-198)$$

$$yColBr = yPb + nPbH \quad (8-199)$$

- If $yPb >> CtbLog2SizeY$ is equal to $yColBr >> CtbLog2SizeY$, $yColBr$ is less than $pic_height_in_luma_samples$ and $xColBr$ is less than $pic_width_in_luma_samples$, the following applies:
 - The variable colPb specifies the luma prediction block covering the modified location given by $((xColBr >> 4) << 4, (yColBr >> 4) << 4)$ inside the collocated picture specified by ColPic.
 - The luma location (xColPb, yColPb) is set equal to the top-left sample of the collocated luma prediction block specified by colPb relative to the top-left luma sample of the collocated picture specified by ColPic.
 - The derivation process for collocated motion vectors as specified in clause 8.5.3.2.9 is invoked with currPb, colPb, (xColPb, yColPb) and refIdxLX as inputs, and the output is assigned to mvLXCol and availableFlagLXCol.
 - Otherwise, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.
- 2. When availableFlagLXCol is equal to 0, the central collocated motion vector is derived as follows:

$$xColCtr = xPb + (nPbW >> 1) \quad (8-200)$$

$$yColCtr = yPb + (nPbH \gg 1) \quad (8-201)$$

- The variable colPb specifies the luma prediction block covering the modified location given by $((xColCtr \gg 4) \ll 4, (yColCtr \gg 4) \ll 4)$ inside the collocated picture specified by ColPic.
- The luma location $(xColPb, yColPb)$ is set equal to the top-left sample of the collocated luma prediction block specified by colPb relative to the top-left luma sample of the collocated picture specified by ColPic.
- The derivation process for collocated motion vectors as specified in clause 8.5.3.2.9 is invoked with currPb, colPb, $(xColPb, yColPb)$ and refIdxLX as inputs, and the output is assigned to mvLXCol and availableFlagLXCol.

8.5.3.2.9 Derivation process for collocated motion vectors

Inputs to this process are:

- a variable currPb specifying the current prediction block,
- a variable colPb specifying the collocated prediction block inside the collocated picture specified by ColPic,
- a luma location $(xColPb, yColPb)$ specifying the top-left sample of the collocated luma prediction block specified by colPb relative to the top-left luma sample of the collocated picture specified by ColPic,
- a reference index refIdxLX, with X being 0 or 1.

Outputs of this process are:

- the motion vector prediction mvLXCol,
- the availability flag availableFlagLXCol.

The variable currPic specifies the current picture.

The arrays predFlagL0Col[x][y], mvL0Col[x][y] and refIdxL0Col[x][y] are set equal to PredFlagL0[x][y], MvL0[x][y] and RefIdxL0[x][y], respectively, of the collocated picture specified by ColPic, and the arrays predFlagL1Col[x][y], mvL1Col[x][y] and refIdxL1Col[x][y] are set equal to PredFlagL1[x][y], MvL1[x][y] and RefIdxL1[x][y], respectively, of the collocated picture specified by ColPic.

The variables mvLXCol and availableFlagLXCol are derived as follows:

- If colPb is coded in an intra prediction mode, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.
- Otherwise, the motion vector mvCol, the reference index refIdxCol and the reference list identifier listCol are derived as follows:
 - If predFlagL0Col[xColPb][yColPb] is equal to 0, mvCol, refIdxCol and listCol are set equal to mvL1Col[xColPb][yColPb], refIdxL1Col[xColPb][yColPb] and L1, respectively.
 - Otherwise, if predFlagL0Col[xColPb][yColPb] is equal to 1 and predFlagL1Col[xColPb][yColPb] is equal to 0, mvCol, refIdxCol and listCol are set equal to mvL0Col[xColPb][yColPb], refIdxL0Col[xColPb][yColPb] and L0, respectively.
 - Otherwise (predFlagL0Col[xColPb][yColPb] is equal to 1 and predFlagL1Col[xColPb][yColPb] is equal to 1), the following assignments are made:
 - If NoBackwardPredFlag is equal to 1, mvCol, refIdxCol and listCol are set equal to mvLXCol[xColPb][yColPb], refIdxLXCol[xColPb][yColPb] and LX, respectively.
 - Otherwise, mvCol, refIdxCol and listCol are set equal to mvLNCOL[xColPb][yColPb], refIdxLNCOL[xColPb][yColPb] and LN, respectively, with N being the value of collocated_from_l0_flag.

And mvLXCol and availableFlagLXCol are derived as follows:

- If LongTermRefPic(currPic, currPb, refIdxLX, LX) is not equal to LongTermRefPic(ColPic, colPb, refIdxCol, listCol), both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.
- Otherwise, the variable availableFlagLXCol is set equal to 1, refPicListCol[refIdxCol] is set to be the picture with reference index refIdxCol in the reference picture list listCol of the slice containing prediction block colPb in the collocated picture specified by ColPic, and the following applies:

$$colPocDiff = DiffPicOrderCnt(ColPic, refPicListCol[refIdxCol]) \quad (8-202)$$

$$\text{currPocDiff} = \text{DiffPicOrderCnt}(\text{currPic}, \text{RefPicListX}[\text{refIdxLX}]) \quad (8-203)$$

- If $\text{RefPicListX}[\text{refIdxLX}]$ is a long-term reference picture, or colPocDiff is equal to currPocDiff , mvLXCol is derived as follows:

$$\text{mvLXCol} = \text{mvCol} \quad (8-204)$$

- Otherwise, mvLXCol is derived as a scaled version of the motion vector mvCol as follows:

$$\text{tx} = (16384 + (\text{Abs}(\text{td}) \gg 1)) / \text{td} \quad (8-205)$$

$$\text{distScaleFactor} = \text{Clip3}(-4096, 4095, (\text{tb} * \text{tx} + 32) \gg 6) \quad (8-206)$$

$$\text{mvLXCol} = \text{Clip3}(-32768, 32767, \text{Sign}(\text{distScaleFactor} * \text{mvCol}) * ((\text{Abs}(\text{distScaleFactor} * \text{mvCol}) + 127) \gg 8)) \quad (8-207)$$

where td and tb are derived as follows:

$$\text{td} = \text{Clip3}(-128, 127, \text{colPocDiff}) \quad (8-208)$$

$$\text{tb} = \text{Clip3}(-128, 127, \text{currPocDiff}) \quad (8-209)$$

8.5.3.2.10 Derivation process for chroma motion vectors

This process is invoked when ChromaArrayType is not equal to 0.

Input to this process is a luma motion vector mvLX .

Output of this process is a chroma motion vector mvCLX .

A chroma motion vector is derived from the corresponding luma motion vector.

For the derivation of the chroma motion vector mvCLX , the following applies:

$$\text{mvCLX}[0] = \text{mvLX}[0] * 2 / \text{SubWidthC} \quad (8-210)$$

$$\text{mvCLX}[1] = \text{mvLX}[1] * 2 / \text{SubHeightC} \quad (8-211)$$

8.5.3.3 Decoding process for inter prediction samples

8.5.3.3.1 General

Inputs to this process are:

- a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a luma location (xBl, yBl) specifying the top-left sample of the current luma prediction block relative to the top-left sample of the current luma coding block,
- a variable nCbS specifying the size of the current luma coding block,
- two variables nPbW and nPbH specifying the width and the height of the luma prediction block,
- the luma motion vectors mvL0 and mvL1 ,
- when ChromaArrayType is not equal to 0, the chroma motion vectors mvCL0 and mvCL1 ,
- the reference indices refIdxL0 and refIdxL1 ,
- the prediction list utilization flags, predFlagL0 , and predFlagL1 .

Outputs of this process are:

- an $(\text{nCbSL}) \times (\text{nCbSL})$ array predSamples_L of luma prediction samples, where nCbSL is derived as specified below,
- when ChromaArrayType is not equal to 0, an $(\text{nCbSw}_C) \times (\text{nCbSh}_C)$ array predSamples_{Cb} of chroma prediction samples for the component Cb , where nCbSw_C and nCbSh_C are derived as specified below,

- when ChromaArrayType is not equal to 0, an $(nCbSw_C) \times (nCbSh_C)$ array $predSamples_{Cr}$ of chroma prediction samples for the component Cr, where $nCbSw_C$ and $nCbSh_C$ are derived as specified below.

The variable $nCbS_L$ is set equal to $nCbS$. When ChromaArrayType is not equal to 0, the variable $nCbSw_C$ is set equal to $nCbS / SubWidthC$ and the variable $nCbSh_C$ is set equal to $nCbS / SubHeightC$.

Let $predSamplesL0_L$ and $predSamplesL1_L$ be $(nPbW) \times (nPbH)$ arrays of predicted luma sample values and, when ChromaArrayType is not equal to 0, $predSamplesL0_{Cb}$, $predSamplesL1_{Cb}$, $predSamplesL0_{Cr}$ and $predSamplesL1_{Cr}$ be $(nPbW / SubWidthC) \times (nPbH / SubHeightC)$ arrays of predicted chroma sample values.

For X being each of 0 and 1, when $predFlagLX$ is equal to 1, the following applies:

- The reference picture consisting of an ordered two-dimensional array $refPicLX_L$ of luma samples and, when ChromaArrayType is not equal to 0, two ordered two-dimensional arrays $refPicLX_{Cb}$ and $refPicLX_{Cr}$ of chroma samples is derived by invoking the process specified in clause 8.5.3.3.2 with $refIdxLX$ as input.
- The array $predSamplesLX_L$ and, when ChromaArrayType is not equal to 0, the arrays $predSamplesLX_{Cb}$ and $predSamplesLX_{Cr}$ are derived by invoking the fractional sample interpolation process specified in clause 8.5.3.3.3 with the luma locations (xCb, yCb) and (xBl, yBl) , the luma prediction block width $nPbW$, the luma prediction block height $nPbH$, the motion vectors $mvLX$ and, when ChromaArrayType is not equal to 0, $mvCLX$, and the reference arrays $refPicLX_L$, $refPicLX_{Cb}$, and $refPicLX_{Cr}$ as inputs.

The prediction samples inside the current luma prediction block, $predSamples_L[x_L + xBl][y_L + yBl]$ with $x_L = 0..nPbW - 1$ and $y_L = 0..nPbH - 1$, are derived by invoking the weighted sample prediction process specified in clause 8.5.3.3.4 with the prediction block width $nPbW$, the prediction block height $nPbH$ and the sample arrays $predSamplesL0_L$ and $predSamplesL1_L$, and the variables $predFlagL0$, $predFlagL1$, $refIdxL0$, $refIdxL1$ and $cIdx$ equal to 0 as inputs.

When ChromaArrayType is not equal to 0, the prediction samples inside the current chroma component Cb prediction block, $predSamples_{Cb}[x_C + xBl / SubWidthC][y_C + yBl / SubHeightC]$ with $x_C = 0..nPbW / SubWidthC - 1$ and $y_C = 0..nPbH / SubHeightC - 1$, are derived by invoking the weighted sample prediction process specified in clause 8.5.3.3.4 with the prediction block width $nPbW$ set equal to $nPbW / SubWidthC$, the prediction block height $nPbH$ set equal to $nPbH / SubHeightC$, the sample arrays $predSamplesL0_{Cb}$ and $predSamplesL1_{Cb}$, and the variables $predFlagL0$, $predFlagL1$, $refIdxL0$, $refIdxL1$ and $cIdx$ equal to 1 as inputs.

When ChromaArrayType is not equal to 0, the prediction samples inside the current chroma component Cr prediction block, $predSamples_{Cr}[x_C + xBl / SubWidthC][y_C + yBl / SubHeightC]$ with $x_C = 0..nPbW / SubWidthC - 1$ and $y_C = 0..nPbH / SubHeightC - 1$, are derived by invoking the weighted sample prediction process specified in clause 8.5.3.3.4 with the prediction block width $nPbW$ set equal to $nPbW / SubWidthC$, the prediction block height $nPbH$ set equal to $nPbH / SubHeightC$, the sample arrays $predSamplesL0_{Cr}$ and $predSamplesL1_{Cr}$, and the variables $predFlagL0$, $predFlagL1$, $refIdxL0$, $refIdxL1$ and $cIdx$ equal to 2 as inputs.

8.5.3.3.2 Reference picture selection process

Input to this process is a reference index $refIdxLX$.

Output of this process is a reference picture consisting of a two-dimensional array of luma samples $refPicLX_L$ and, when ChromaArrayType is not equal to 0, two two-dimensional arrays of chroma samples $refPicLX_{Cb}$ and $refPicLX_{Cr}$.

The output reference picture $RefPicListX[refIdxLX]$ consists of a $pic_width_in_luma_samples$ by $pic_height_in_luma_samples$ array of luma samples $refPicLX_L$ and, when ChromaArrayType is not equal to 0, two $PicWidthInSamplesC$ by $PicHeightInSamplesC$ arrays of chroma samples $refPicLX_{Cb}$ and $refPicLX_{Cr}$.

The reference picture sample arrays $refPicLX_L$, $refPicLX_{Cb}$ and $refPicLX_{Cr}$ correspond to decoded sample arrays S_L , S_{Cb} and S_{Cr} derived in clause 8.7 for a previously-decoded picture.

8.5.3.3.3 Fractional sample interpolation process

8.5.3.3.3.1 General

Inputs to this process are:

- a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture
- a luma location (xBl, yBl) specifying the top-left sample of the current luma prediction block relative to the top-left sample of the current luma coding block
- two variables $nPbW$ and $nPbH$ specifying the width and the height of the luma prediction block
- a luma motion vector $mvLX$ given in quarter-luma-sample units

- when ChromaArrayType is not equal to 0, a chroma motion vector mvCLX given in eighth-chroma-sample units
- the selected reference picture sample array refPicLX_L and, when ChromaArrayType is not equal to 0, the arrays refPicLX_{Cb} and refPicLX_{Cr}.

Outputs of this process are:

- an (nPbW)x(nPbH) array predSamplesLX_L of prediction luma sample values
- when ChromaArrayType is not equal to 0, two (nPbW / SubWidthC)x(nPbH / SubHeightC) arrays predSamplesLX_{Cb} and predSamplesLX_{Cr} of prediction chroma sample values.

The location (xPb, yPb) given in full-sample units of the upper-left luma samples of the current prediction block relative to the upper-left luma sample location of the given reference sample arrays is derived as follows:

$$xPb = xCb + xBl \quad (8-212)$$

$$yPb = yCb + yBl \quad (8-213)$$

Let (xInt_L, yInt_L) be a luma location given in full-sample units and (xFrac_L, yFrac_L) be an offset given in quarter-sample units. These variables are used only in this clause for specifying fractional-sample locations inside the reference sample arrays refPicLX_L, refPicLX_{Cb} and refPicLX_{Cr}.

For each luma sample location (x_L = 0..nPbW – 1, y_L = 0..nPbH – 1) inside the prediction luma sample array predSamplesLX_L, the corresponding prediction luma sample value predSamplesLX_L[x_L][y_L] is derived as follows:

- The variables xInt_L, yInt_L, xFrac_L and yFrac_L are derived as follows:

$$xInt_L = xPb + (mvLX[0] \gg 2) + x_L \quad (8-214)$$

$$yInt_L = yPb + (mvLX[1] \gg 2) + y_L \quad (8-215)$$

$$xFrac_L = mvLX[0] \& 3 \quad (8-216)$$

$$yFrac_L = mvLX[1] \& 3 \quad (8-217)$$

- The prediction luma sample value predSamplesLX_L[x_L][y_L] is derived by invoking the process specified in clause 8.5.3.3.2 with (xInt_L, yInt_L), (xFrac_L, yFrac_L) and refPicLX_L as inputs.

When ChromaArrayType is not equal to 0, the following applies.

Let (xInt_C, yInt_C) be a chroma location given in full-sample units and (xFrac_C, yFrac_C) be an offset given in one-eighth sample units. These variables are used only in this clause for specifying general fractional-sample locations inside the reference sample arrays refPicLX_{Cb} and refPicLX_{Cr}.

For each chroma sample location (x_C = 0..nPbW / SubWidthC – 1, y_C = 0..nPbH / SubHeightC – 1) inside the prediction chroma sample arrays predSamplesLX_{Cb} and predSamplesLX_{Cr}, the corresponding prediction chroma sample values predSamplesLX_{Cb}[x_C][y_C] and predSamplesLX_{Cr}[x_C][y_C] are derived as follows:

- The variables xInt_C, yInt_C, xFrac_C and yFrac_C are derived as follows:

$$xInt_C = (xPb / SubWidthC) + (mvCLX[0] \gg 3) + x_C \quad (8-218)$$

$$yInt_C = (yPb / SubHeightC) + (mvCLX[1] \gg 3) + y_C \quad (8-219)$$

$$xFrac_C = mvCLX[0] \& 7 \quad (8-220)$$

$$yFrac_C = mvCLX[1] \& 7 \quad (8-221)$$

- The prediction sample value predSamplesLX_{Cb}[x_C][y_C] is derived by invoking the process specified in clause 8.5.3.3.3 with (xInt_C, yInt_C), (xFrac_C, yFrac_C) and refPicLX_{Cb} as inputs.
- The prediction sample value predSamplesLX_{Cr}[x_C][y_C] is derived by invoking the process specified in clause 8.5.3.3.3 with (xInt_C, yInt_C), (xFrac_C, yFrac_C) and refPicLX_{Cr} as inputs.

8.5.3.3.2 Luma sample interpolation process

Inputs to this process are:

- a luma location in full-sample units (x_{Int_L} , y_{Int_L}),
- a luma location in fractional-sample units (x_{Frac_L} , y_{Frac_L}),
- the luma reference sample array $refPicLX_L$.

Output of this process is a predicted luma sample value $predSampleLX_L$

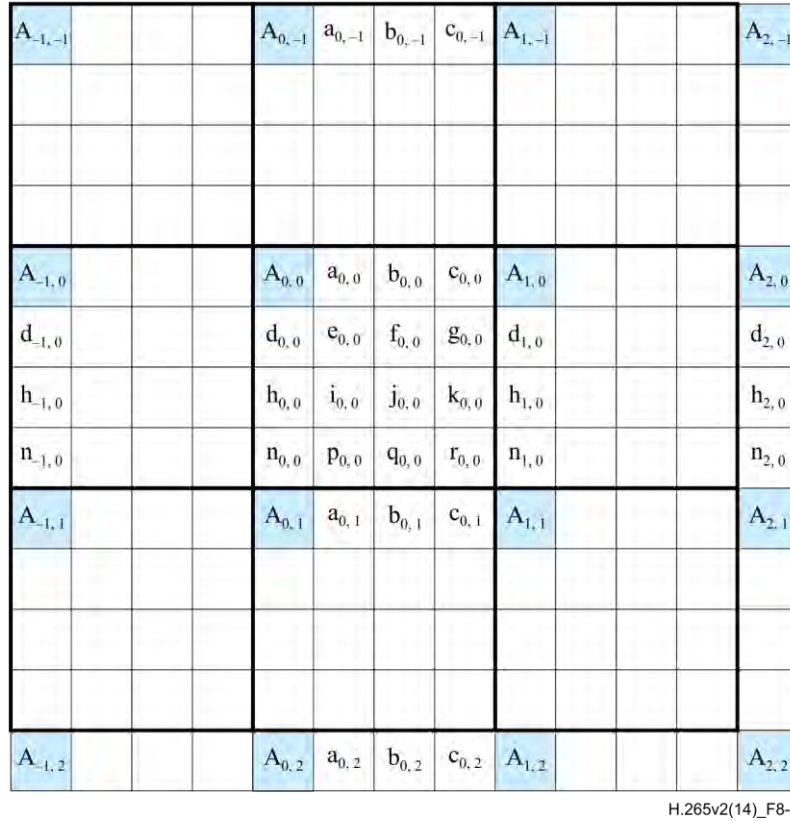


Figure 8-4 – Integer samples (shaded blocks with upper-case letters) and fractional sample positions (un-shaded blocks with lower-case letters) for quarter sample luma interpolation

In Figure 8-4, the positions labelled with upper-case letters $A_{i,j}$ within shaded blocks represent luma samples at full-sample locations inside the given two-dimensional array $refPicLX_L$ of luma samples. These samples may be used for generating the predicted luma sample value $predSampleLX_L$. The locations ($x_{A_{i,j}}$, $y_{A_{i,j}}$) for each of the corresponding luma samples $A_{i,j}$ inside the given array $refPicLX_L$ of luma samples are derived as follows:

$$x_{A_{i,j}} = \text{Clip3}(0, \text{pic_width_in_luma_samples} - 1, x_{Int_L} + i) \quad (8-222)$$

$$y_{A_{i,j}} = \text{Clip3}(0, \text{pic_height_in_luma_samples} - 1, y_{Int_L} + j) \quad (8-223)$$

The positions labelled with lower-case letters within un-shaded blocks represent luma samples at quarter-luma-sample fractional locations. The luma location offset in fractional-sample units (x_{Frac_L} , y_{Frac_L}) specifies which of the generated luma samples at full-sample and fractional-sample locations is assigned to the predicted luma sample value $predSampleLX_L$. This assignment is as specified in Table 8-8. The value of $predSampleLX_L$ is the output.

The variables $shift1$, $shift2$ and $shift3$ are derived as follows:

- The variable $shift1$ is set equal to $\text{Min}(4, \text{BitDepth}_Y - 8)$, the variable $shift2$ is set equal to 6 and the variable $shift3$ is set equal to $\text{Max}(2, 14 - \text{BitDepth}_Y)$.

Given the luma samples $A_{i,j}$ at full-sample locations ($x_{A_{i,j}}$, $y_{A_{i,j}}$), the luma samples $a_{0,0}$ to $r_{0,0}$ at fractional sample positions are derived as follows:

- The samples labelled $a_{0,0}$, $b_{0,0}$, $c_{0,0}$, $d_{0,0}$, $h_{0,0}$ and $n_{0,0}$ are derived by applying an 8-tap filter to the nearest integer position samples as follows:

$$a_{0,0} = (-A_{-3,0} + 4 * A_{-2,0} - 10 * A_{-1,0} + 58 * A_{0,0} + 17 * A_{1,0} - 5 * A_{2,0} + A_{3,0}) \gg shift1 \quad (8-224)$$

$$b_{0,0} = (-A_{-3,0} + 4 * A_{-2,0} - 11 * A_{-1,0} + 40 * A_{0,0} + 40 * A_{1,0} - 11 * A_{2,0} + 4 * A_{3,0} - A_{4,0}) >> \text{shift1} \quad (8-225)$$

$$c_{0,0} = (A_{-2,0} - 5 * A_{-1,0} + 17 * A_{0,0} + 58 * A_{1,0} - 10 * A_{2,0} + 4 * A_{3,0} - A_{4,0}) >> \text{shift1} \quad (8-226)$$

$$d_{0,0} = (-A_{0,-3} + 4 * A_{0,-2} - 10 * A_{0,-1} + 58 * A_{0,0} + 17 * A_{0,1} - 5 * A_{0,2} + A_{0,3}) >> \text{shift1} \quad (8-227)$$

$$h_{0,0} = (-A_{0,-3} + 4 * A_{0,-2} - 11 * A_{0,-1} + 40 * A_{0,0} + 40 * A_{0,1} - 11 * A_{0,2} + 4 * A_{0,3} - A_{0,4}) >> \text{shift1} \quad (8-228)$$

$$n_{0,0} = (A_{0,-2} - 5 * A_{0,-1} + 17 * A_{0,0} + 58 * A_{0,1} - 10 * A_{0,2} + 4 * A_{0,3} - A_{0,4}) >> \text{shift1} \quad (8-229)$$

- The samples labelled $e_{0,0}$, $i_{0,0}$, $p_{0,0}$, $f_{0,0}$, $j_{0,0}$, $q_{0,0}$, $g_{0,0}$, $k_{0,0}$ and $r_{0,0}$ are derived by applying an 8-tap filter to the samples $a_{0,i}$, $b_{0,i}$ and $c_{0,i}$ with $i = -3..4$ in the vertical direction as follows:

$$e_{0,0} = (-a_{0,-3} + 4 * a_{0,-2} - 10 * a_{0,-1} + 58 * a_{0,0} + 17 * a_{0,1} - 5 * a_{0,2} + a_{0,3}) >> \text{shift2} \quad (8-230)$$

$$i_{0,0} = (-a_{0,-3} + 4 * a_{0,-2} - 11 * a_{0,-1} + 40 * a_{0,0} + 40 * a_{0,1} - 11 * a_{0,2} + 4 * a_{0,3} - a_{0,4}) >> \text{shift2} \quad (8-231)$$

$$p_{0,0} = (a_{0,-2} - 5 * a_{0,-1} + 17 * a_{0,0} + 58 * a_{0,1} - 10 * a_{0,2} + 4 * a_{0,3} - a_{0,4}) >> \text{shift2} \quad (8-232)$$

$$f_{0,0} = (-b_{0,-3} + 4 * b_{0,-2} - 10 * b_{0,-1} + 58 * b_{0,0} + 17 * b_{0,1} - 5 * b_{0,2} + b_{0,3}) >> \text{shift2} \quad (8-233)$$

$$j_{0,0} = (-b_{0,-3} + 4 * b_{0,-2} - 11 * b_{0,-1} + 40 * b_{0,0} + 40 * b_{0,1} - 11 * b_{0,2} + 4 * b_{0,3} - b_{0,4}) >> \text{shift2} \quad (8-234)$$

$$q_{0,0} = (b_{0,-2} - 5 * b_{0,-1} + 17 * b_{0,0} + 58 * b_{0,1} - 10 * b_{0,2} + 4 * b_{0,3} - b_{0,4}) >> \text{shift2} \quad (8-235)$$

$$g_{0,0} = (-c_{0,-3} + 4 * c_{0,-2} - 10 * c_{0,-1} + 58 * c_{0,0} + 17 * c_{0,1} - 5 * c_{0,2} + c_{0,3}) >> \text{shift2} \quad (8-236)$$

$$k_{0,0} = (-c_{0,-3} + 4 * c_{0,-2} - 11 * c_{0,-1} + 40 * c_{0,0} + 40 * c_{0,1} - 11 * c_{0,2} + 4 * c_{0,3} - c_{0,4}) >> \text{shift2} \quad (8-237)$$

$$r_{0,0} = (c_{0,-2} - 5 * c_{0,-1} + 17 * c_{0,0} + 58 * c_{0,1} - 10 * c_{0,2} + 4 * c_{0,3} - c_{0,4}) >> \text{shift2} \quad (8-238)$$

Table 8-8 – Assignment of the luma prediction sample predSampleLXL

xFracL	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
yFracL	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
predSampleLXL	A << shift3	d	h	n	a	e	i	p	b	f	j	q	c	g	k	r

8.5.3.3.3 Chroma sample interpolation process

This process is only invoked when ChromaArrayType is not equal to 0.

Inputs to this process are:

- a chroma location in full-sample units ($xInt_C$, $yInt_C$),
- a chroma location in eighth fractional-sample units ($xFrac_C$, $yFrac_C$),
- the chroma reference sample array $refPicLXC$.

Output of this process is a predicted chroma sample value $predSampleLXC$

	ha _{0,-1}	hb _{0,-1}	hc _{0,-1}	hd _{0,-1}	he _{0,-1}	hf _{0,-1}	hg _{0,-1}	hh _{0,-1}	
ah _{-1,0}	B _{0,0}	ab _{0,0}	ac _{0,0}	ad _{0,0}	ae _{0,0}	af _{0,0}	ag _{0,0}	ah _{0,0}	B _{1,0}
bh _{-1,0}	ba _{0,0}	bb _{0,0}	bc _{0,0}	bd _{0,0}	be _{0,0}	bf _{0,0}	bg _{0,0}	bh _{0,0}	ba _{1,0}
ch _{-1,0}	ca _{0,0}	cb _{0,0}	cc _{0,0}	cd _{0,0}	ce _{0,0}	cf _{0,0}	cg _{0,0}	ch _{0,0}	ca _{1,0}
dh _{-1,0}	da _{0,0}	db _{0,0}	dc _{0,0}	dd _{0,0}	de _{0,0}	df _{0,0}	dg _{0,0}	dh _{0,0}	da _{1,0}
eh _{-1,0}	ea _{0,0}	eb _{0,0}	ec _{0,0}	ed _{0,0}	ee _{0,0}	ef _{0,0}	eg _{0,0}	eh _{0,0}	ea _{1,0}
fh _{-1,0}	fa _{0,0}	fb _{0,0}	fc _{0,0}	fd _{0,0}	fe _{0,0}	ff _{0,0}	fg _{0,0}	fh _{0,0}	fa _{1,0}
gh _{-1,0}	ga _{0,0}	gb _{0,0}	gc _{0,0}	gd _{0,0}	ge _{0,0}	gf _{0,0}	gg _{0,0}	gh _{0,0}	ga _{1,0}
hh _{-1,0}	ha _{0,0}	hb _{0,0}	hc _{0,0}	hd _{0,0}	he _{0,0}	hf _{0,0}	hg _{0,0}	hh _{0,0}	ha _{1,0}
	B _{0,1}	ab _{0,1}	ac _{0,1}	ad _{0,1}	ae _{0,1}	af _{0,1}	ag _{0,1}	ah _{0,1}	B _{1,1}

H.265v2(14)_F8-5

Figure 8-5 – Integer samples (shaded blocks with upper-case letters) and fractional sample positions (un-shaded blocks with lower-case letters) for eighth sample chroma interpolation

In Figure 8-5, the positions labelled with upper-case letters $B_{i,j}$ within shaded blocks represent chroma samples at full-sample locations inside the given two-dimensional array refPicLXC of chroma samples. These samples may be used for generating the predicted chroma sample value predSampleLXC . The locations $(x_{B_{i,j}}, y_{B_{i,j}})$ for each of the corresponding chroma samples $B_{i,j}$ inside the given array refPicLXC of chroma samples are derived as follows:

$$x_{B_{i,j}} = \text{Clip3}(0, (\text{pic_width_in_luma_samples} / \text{SubWidthC}) - 1, x_{\text{IntC}} + i) \quad (8-239)$$

$$y_{B_{i,j}} = \text{Clip3}(0, (\text{pic_height_in_luma_samples} / \text{SubHeightC}) - 1, y_{\text{IntC}} + j) \quad (8-240)$$

The positions labelled with lower-case letters within un-shaded blocks represent chroma samples at eighth-pel sample fractional locations. The chroma location offset in fractional-sample units $(x_{\text{FracC}}, y_{\text{FracC}})$ specifies which of the generated chroma samples at full-sample and fractional-sample locations is assigned to the predicted chroma sample value predSampleLXC . This assignment is as specified in Table 8-9. The output is the value of predSampleLXC .

The variables shift1 , shift2 and shift3 are derived as follows:

- The variable shift1 is set equal to $\text{Min}(4, \text{BitDepthC} - 8)$, the variable shift2 is set equal to 6 and the variable shift3 is set equal to $\text{Max}(2, 14 - \text{BitDepthC})$.

Given the chroma samples $B_{i,j}$ at full-sample locations $(x_{B_{i,j}}, y_{B_{i,j}})$, the chroma samples $ab_{0,0}$ to $hh_{0,0}$ at fractional sample positions are derived as follows:

- The samples labelled $ab_{0,0}$, $ac_{0,0}$, $ad_{0,0}$, $ae_{0,0}$, $af_{0,0}$, $ag_{0,0}$ and $ah_{0,0}$ are derived by applying a 4-tap filter to the nearest integer position samples as follows:

$$ab_{0,0} = (-2 * B_{-1,0} + 58 * B_{0,0} + 10 * B_{1,0} - 2 * B_{2,0}) \gg \text{shift1} \quad (8-241)$$

$$ac_{0,0} = (-4 * B_{-1,0} + 54 * B_{0,0} + 16 * B_{1,0} - 2 * B_{2,0}) \gg \text{shift1} \quad (8-242)$$

$$ad_{0,0} = (-6 * B_{-1,0} + 46 * B_{0,0} + 28 * B_{1,0} - 4 * B_{2,0}) \gg \text{shift1} \quad (8-243)$$

$$ae_{0,0} = (-4 * B_{-1,0} + 36 * B_{0,0} + 36 * B_{1,0} - 4 * B_{2,0}) \gg \text{shift1} \quad (8-244)$$

$$af_{0,0} = (-4 * B_{-1,0} + 28 * B_{0,0} + 46 * B_{1,0} - 6 * B_{2,0}) \gg \text{shift1} \quad (8-245)$$

$$ag_{0,0} = (-2 * B_{-1,0} + 16 * B_{0,0} + 54 * B_{1,0} - 4 * B_{2,0}) \gg \text{shift1} \quad (8-246)$$

$$ah_{0,0} = (-2 * B_{-1,0} + 10 * B_{0,0} + 58 * B_{1,0} - 2 * B_{2,0}) \gg \text{shift1} \quad (8-247)$$

- The samples labelled $ba_{0,0}$, $ca_{0,0}$, $da_{0,0}$, $ea_{0,0}$, $fa_{0,0}$, $ga_{0,0}$ and $ha_{0,0}$ are derived by applying a 4-tap filter to the nearest integer position samples as follows:

$$ba_{0,0} = (-2 * B_{0,-1} + 58 * B_{0,0} + 10 * B_{0,1} - 2 * B_{0,2}) \gg \text{shift1} \quad (8-248)$$

$$ca_{0,0} = (-4 * B_{0,-1} + 54 * B_{0,0} + 16 * B_{0,1} - 2 * B_{0,2}) \gg \text{shift1} \quad (8-249)$$

$$da_{0,0} = (-6 * B_{0,-1} + 46 * B_{0,0} + 28 * B_{0,1} - 4 * B_{0,2}) \gg \text{shift1} \quad (8-250)$$

$$ea_{0,0} = (-4 * B_{0,-1} + 36 * B_{0,0} + 36 * B_{0,1} - 4 * B_{0,2}) \gg \text{shift1} \quad (8-251)$$

$$fa_{0,0} = (-4 * B_{0,-1} + 28 * B_{0,0} + 46 * B_{0,1} - 6 * B_{0,2}) \gg \text{shift1} \quad (8-252)$$

$$ga_{0,0} = (-2 * B_{0,-1} + 16 * B_{0,0} + 54 * B_{0,1} - 4 * B_{0,2}) \gg \text{shift1} \quad (8-253)$$

$$ha_{0,0} = (-2 * B_{0,-1} + 10 * B_{0,0} + 58 * B_{0,1} - 2 * B_{0,2}) \gg \text{shift1} \quad (8-254)$$

- The samples labelled $bX_{0,0}$, $cX_{0,0}$, $dX_{0,0}$, $eX_{0,0}$, $fX_{0,0}$, $gX_{0,0}$ and $hX_{0,0}$ for X being replaced by b, c, d, e, f, g and h, respectively, are derived by applying a 4-tap filter to the intermediate values $aX_{0,i}$ with $i = -1..2$ in the vertical direction as follows:

$$bX_{0,0} = (-2 * aX_{0,-1} + 58 * aX_{0,0} + 10 * aX_{0,1} - 2 * aX_{0,2}) \gg \text{shift2} \quad (8-255)$$

$$cX_{0,0} = (-4 * aX_{0,-1} + 54 * aX_{0,0} + 16 * aX_{0,1} - 2 * aX_{0,2}) \gg \text{shift2} \quad (8-256)$$

$$dX_{0,0} = (-6 * aX_{0,-1} + 46 * aX_{0,0} + 28 * aX_{0,1} - 4 * aX_{0,2}) \gg \text{shift2} \quad (8-257)$$

$$eX_{0,0} = (-4 * aX_{0,-1} + 36 * aX_{0,0} + 36 * aX_{0,1} - 4 * aX_{0,2}) \gg \text{shift2} \quad (8-258)$$

$$fX_{0,0} = (-4 * aX_{0,-1} + 28 * aX_{0,0} + 46 * aX_{0,1} - 6 * aX_{0,2}) \gg \text{shift2} \quad (8-259)$$

$$gX_{0,0} = (-2 * aX_{0,-1} + 16 * aX_{0,0} + 54 * aX_{0,1} - 4 * aX_{0,2}) \gg \text{shift2} \quad (8-260)$$

$$hX_{0,0} = (-2 * aX_{0,-1} + 10 * aX_{0,0} + 58 * aX_{0,1} - 2 * aX_{0,2}) \gg \text{shift2} \quad (8-261)$$

Table 8-9 – Assignment of the chroma prediction sample predSampleLX_C for (X, Y) being replaced by (1, b), (2, c), (3, d), (4, e), (5, f), (6, g) and (7, h), respectively

xFracC	0	0	0	0	0	0	0	0
yFracC	0	1	2	3	4	5	6	7
predSampleLX_C	B << shift3	ba	ca	da	ea	fa	ga	ha
xFracC	X	X	X	X	X	X	X	X
yFracC	0	1	2	3	4	5	6	7
predSampleLX_C	aY	bY	cY	dY	eY	fY	gY	hY

8.5.3.3.4 Weighted sample prediction process

8.5.3.3.4.1 General

Inputs to this process are:

- two variables $nPbW$ and $nPbH$ specifying the width and the height of the current prediction block,
- two $(nPbW) \times (nPbH)$ arrays predSamplesL0 and predSamplesL1 ,
- the prediction list utilization flags, predFlagL0 and predFlagL1 ,
- the reference indices refIdxL0 and refIdxL1 ,
- a variable $cIdx$ specifying colour component index.

Output of this process is the $(nPbW) \times (nPbH)$ array pbSamples of prediction sample values.

The variable bitDepth is derived as follows:

- If cIdx is equal to 0, bitDepth is set equal to BitDepth_Y.
- Otherwise, bitDepth is set equal to BitDepth_C.

The variable weightedPredFlag is derived as follows:

- If slice_type is equal to P, weightedPredFlag is set equal to weighted_pred_flag.
- Otherwise (slice_type is equal to B), weightedPredFlag is set equal to weighted_bipred_flag.

The following applies:

- If weightedPredFlag is equal to 0, the array pbSamples of the prediction samples is derived by invoking the default weighted sample prediction process as specified in clause 8.5.3.3.4.2 with the prediction block width nPbW, the prediction block height nPbH, two $(nPbW) \times (nPbH)$ arrays predSamplesL0 and predSamplesL1, the prediction list utilization flags predFlagL0 and predFlagL1 and the bit depth bitDepth as inputs.
- Otherwise (weightedPredFlag is equal to 1), the array pbSamples of the prediction samples is derived by invoking the weighted sample prediction process as specified in clause 8.5.3.3.4.3 with the prediction block width nPbW, the prediction block height nPbH, two $(nPbW) \times (nPbH)$ arrays predSamplesL0 and predSamplesL1, the prediction list utilization flags predFlagL0 and predFlagL1, the reference indices refIdxL0 and refIdxL1, the colour component index cIdx and the bit depth bitDepth as inputs.

8.5.3.3.4.2 Default weighted sample prediction process

Inputs to this process are:

- two variables nPbW and nPbH specifying the width and the height of the current prediction block,
- two $(nPbW) \times (nPbH)$ arrays predSamplesL0 and predSamplesL1,
- the prediction list utilization flags, predFlagL0, and predFlagL1,
- a bit depth of samples, bitDepth.

Output of this process is the $(nPbW) \times (nPbH)$ array pbSamples of prediction sample values.

Variables shift1, shift2, offset1 and offset2 are derived as follows:

- The variable shift1 is set equal to $\text{Max}(2, 14 - \text{bitDepth})$ and the variable shift2 is set equal to $\text{Max}(3, 15 - \text{bitDepth})$.
- The variable offset1 is set equal to $1 \ll (\text{shift1} - 1)$.
- The variable offset2 is set equal to $1 \ll (\text{shift2} - 1)$.

Depending on the values of predFlagL0 and predFlagL1, the prediction samples pbSamples[x][y] with $x = 0..nPbW - 1$ and $y = 0..nPbH - 1$ are derived as follows:

- If predFlagL0 is equal to 1 and predFlagL1 is equal to 0, the prediction sample values are derived as follows:

$$\text{pbSamples}[x][y] = \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, (\text{predSamplesL0}[x][y] + \text{offset1}) \gg \text{shift1}) \quad (8-262)$$

- Otherwise, if predFlagL0 is equal to 0 and predFlagL1 is equal to 1, the prediction sample values are derived as follows:

$$\text{pbSamples}[x][y] = \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, (\text{predSamplesL1}[x][y] + \text{offset1}) \gg \text{shift1}) \quad (8-263)$$

- Otherwise (predFlagL0 is equal to 1 and predFlagL1 is equal to 1), the prediction sample values are derived as follows:

$$\text{pbSamples}[x][y] = \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, (\text{predSamplesL0}[x][y] + \text{predSamplesL1}[x][y] + \text{offset2}) \gg \text{shift2}) \quad (8-264)$$

8.5.3.3.4.3 Explicit weighted sample prediction process

Inputs to this process are:

- two variables nPbW and nPbH specifying the width and the height of the current prediction block,
- two (nPbW)x(nPbH) arrays predSamplesL0 and predSamplesL1,
- the prediction list utilization flags, predFlagL0 and predFlagL1,
- the reference indices, refIdxL0 and refIdxL1,
- a variable cIdx specifying colour component index,
- a bit depth of samples, bitDepth.

Output of this process is the (nPbW)x(nPbH) array pbSamples of prediction sample values.

The variable shift1 is set equal to $\text{Max}(2, 14 - \text{bitDepth})$.

The variables log2Wd, o0, o1, w0 and w1 are derived as follows:

- If cIdx is equal to 0 for luma samples, the following applies:

$$\text{log2Wd} = \text{luma_log2_weight_denom} + \text{shift1} \quad (8-265)$$

$$\text{w0} = \text{LumaWeightL0}[\text{refIdxL0}] \quad (8-266)$$

$$\text{w1} = \text{LumaWeightL1}[\text{refIdxL1}] \quad (8-267)$$

$$\text{o0} = \text{luma_offset_l0}[\text{refIdxL0}] \ll \text{WpOffsetBdShift}_Y \quad (8-268)$$

$$\text{o1} = \text{luma_offset_l1}[\text{refIdxL1}] \ll \text{WpOffsetBdShift}_Y \quad (8-269)$$

- Otherwise (cIdx is not equal to 0 for chroma samples), the following applies:

$$\text{log2Wd} = \text{ChromaLog2WeightDenom} + \text{shift1} \quad (8-270)$$

$$\text{w0} = \text{ChromaWeightL0}[\text{refIdxL0}][\text{cIdx} - 1] \quad (8-271)$$

$$\text{w1} = \text{ChromaWeightL1}[\text{refIdxL1}][\text{cIdx} - 1] \quad (8-272)$$

$$\text{o0} = \text{ChromaOffsetL0}[\text{refIdxL0}][\text{cIdx} - 1] \ll \text{WpOffsetBdShift}_C \quad (8-273)$$

$$\text{o1} = \text{ChromaOffsetL1}[\text{refIdxL1}][\text{cIdx} - 1] \ll \text{WpOffsetBdShift}_C \quad (8-274)$$

The prediction sample pbSamples[x][y] with $x = 0..nPbW - 1$ and $y = 0..nPbH - 1$ are derived as follows:

- If the predFlagL0 is equal to 1 and predFlagL1 is equal to 0, the prediction sample values are derived as follows:

$$\text{pbSamples}[x][y] = \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, ((\text{predSamplesL0}[x][y] * \text{w0} + 2^{\text{log2Wd}-1}) \gg \text{log2Wd}) + \text{o0}) \quad (8-275)$$

- Otherwise, if the predFlagL0 is equal to 0 and predFlagL1 is equal to 1, the prediction sample values are derived as follows:

$$\text{pbSamples}[x][y] = \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, ((\text{predSamplesL1}[x][y] * \text{w1} + 2^{\text{log2Wd}-1}) \gg \text{log2Wd}) + \text{o1}) \quad (8-276)$$

- Otherwise (predFlagL0 is equal to 1 and predFlagL1 is equal to 1), the prediction sample values are derived as follows:

$$\text{pbSamples}[x][y] = \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, (\text{predSamplesL0}[x][y] * \text{w0} + \text{predSamplesL1}[x][y] * \text{w1} + ((\text{o0} + \text{o1} + 1) \ll \text{log2Wd})) \gg (\text{log2Wd} + 1)) \quad (8-277)$$

8.5.4 Decoding process for the residual signal of coding units coded in inter prediction mode

8.5.4.1 General

Inputs to this process are:

- a luma location (x_{Cb} , y_{Cb}) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a variable $\log_2 CbSize$ specifying the size of the current luma coding block.

Outputs of this process are:

- an $(nCbS_L) \times (nCbS_L)$ array $resSamples_L$ of luma residual samples, where $nCbS_L$ is derived as specified below,
- when $ChromaArrayType$ is not equal to 0, an $(nCbSw_C) \times (nCbSh_C)$ array $resSamples_{Cb}$ of chroma residual samples for the component Cb, where $nCbSw_C$ and $nCbSh_C$ are derived as specified below,
- when $ChromaArrayType$ is not equal to 0, an $(nCbSw_C) \times (nCbSh_C)$ array $resSamples_{Cr}$ of chroma residual samples for the component Cr, where $nCbSw_C$ and $nCbSh_C$ are derived as specified below.

The variable $nCbS_L$ is set equal to $1 \ll \log_2 CbSize$. When $ChromaArrayType$ is not equal to 0, the variable $nCbSw_C$ is set equal to $nCbS_L / SubWidthC$ and the variable $nCbSh_C$ is set equal to $nCbS_L / SubHeightC$.

Let $resSamples_L$ be an $(nCbS_L) \times (nCbS_L)$ array of luma residual samples and, when $ChromaArrayType$ is not equal to 0, let $resSamples_{Cb}$ and $resSamples_{Cr}$ be two $(nCbSw_C) \times (nCbSh_C)$ arrays of chroma residual samples.

Depending on the value of rqt_root_cbf , the following applies:

- If rqt_root_cbf is equal to 0 or $cu_skip_flag[x_{Cb}][y_{Cb}]$ is equal to 1, all samples of the $(nCbS_L) \times (nCbS_L)$ array $resSamples_L$ and, when $ChromaArrayType$ is not equal to 0, all samples of the two $(nCbSw_C) \times (nCbSh_C)$ arrays $resSamples_{Cb}$ and $resSamples_{Cr}$ are set equal to 0.
- Otherwise (rqt_root_cbf is equal to 1), the following ordered steps apply:
 1. The decoding process for luma residual blocks as specified in clause 8.5.4.2 below is invoked with the luma location (x_{Cb} , y_{Cb}), the luma location (x_{B0} , y_{B0}) set equal to (0, 0), the variable $\log_2 TrafoSize$ set equal to $\log_2 CbSize$, the variable $trafoDepth$ set equal to 0, the variable $nCbS$ set equal to $nCbS_L$ and the $(nCbS_L) \times (nCbS_L)$ array $resSamples_L$ as inputs, and the output is a modified version of the $(nCbS_L) \times (nCbS_L)$ array $resSamples_L$.
 2. When $ChromaArrayType$ is not equal to 0, the decoding process for chroma residual blocks as specified in clause 8.5.4.3 is invoked with the luma location (x_{Cb} , y_{Cb}), the luma location (x_{B0} , y_{B0}) set equal to (0, 0), the variable $\log_2 TrafoSize$ set equal to $\log_2 CbSize$, the variable $trafoDepth$ set equal to 0, the variable $cIdx$ set equal to 1, the variable $nCbSw$ set equal to $nCbSw_C$, the variable $nCbSh$ set equal to $nCbSh_C$ and the $(nCbSw_C) \times (nCbSh_C)$ array $resSamples_{Cb}$ as inputs, and the output is a modified version of the $(nCbSw_C) \times (nCbSh_C)$ array $resSamples_{Cb}$.
 3. When $ChromaArrayType$ is not equal to 0, the decoding process for chroma residual blocks as specified in clause 8.5.4.3 is invoked with the luma location (x_{Cb} , y_{Cb}), the luma location (x_{B0} , y_{B0}) set equal to (0, 0), the variable $\log_2 TrafoSize$ set equal to $\log_2 CbSize$, the variable $trafoDepth$ set equal to 0, the variable $cIdx$ set equal to 2, the variable $nCbSw$ set equal to $nCbSw_C$, the variable $nCbSh$ set equal to $nCbSh_C$ and the $(nCbSw_C) \times (nCbSh_C)$ array $resSamples_{Cr}$ as inputs, and the output is a modified version of the $(nCbSw_C) \times (nCbSh_C)$ array $resSamples_{Cr}$.
 4. When $residual_adaptive_colour_transform_enabled_flag$ is equal to 1, the residual modification process for blocks using adaptive colour transform as specified in clause 8.6.8 is invoked with location (x_{Cb} , y_{Cb}), the variable $\log_2 TrafoSize$ set equal to $\log_2 CbSize$, the variable $trafoDepth$ set equal to 0, the variable $resSampleArrayL$ set equal to $resSamples_L$, the variable $resSampleArrayCb$ set equal to $resSamples_{Cb}$ and the variable $resSampleArrayCr$ set equal to $resSamples_{Cr}$ as inputs, and the outputs are modified versions of $resSample_L$, $resSample_{Cb}$ and $resSample_{Cr}$.

8.5.4.2 Decoding process for luma residual blocks

Inputs to this process are:

- a luma location (x_{Cb} , y_{Cb}) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a luma location (x_{B0} , y_{B0}) specifying the top-left sample of the current luma block relative to the top-left sample of the current luma coding block,
- a variable $\log_2 TrafoSize$ specifying the size of the current luma block,
- a variable $trafoDepth$ specifying the hierarchy depth of the current luma block relative to the luma coding block,
- a variable $nCbS$ specifying the size of the current luma coding block,
- an $(nCbS) \times (nCbS)$ array $resSamples$ of luma residual samples.

Output of this process is a modified version of the $(nCbS) \times (nCbS)$ array of luma residual samples.

Depending on the value of $split_transform_flag[xCb + xB0][yCb + yB0][trafoDepth]$, the following applies:

- If $split_transform_flag[xCb + xB0][yCb + yB0][trafoDepth]$ is equal to 1, the following ordered steps apply:
 1. The variables $xB1$ and $yB1$ are derived as follows:
 - The variable $xB1$ is set equal to $xB0 + (1 \ll (\log2TrafoSize - 1))$.
 - The variable $yB1$ is set equal to $yB0 + (1 \ll (\log2TrafoSize - 1))$.
 2. The decoding process for luma residual blocks as specified in this clause is invoked with the luma location (xCb, yCb) , the luma location $(xB0, yB0)$, the variable $\log2TrafoSize$ set equal to $\log2TrafoSize - 1$, the variable $trafoDepth$ set equal to $trafoDepth + 1$, the variable $nCbS$ and the $(nCbS) \times (nCbS)$ array $resSamples$ as inputs, and the output is a modified version of the $(nCbS) \times (nCbS)$ array $resSamples$.
 3. The decoding process for luma residual blocks as specified in this clause is invoked with the luma location (xCb, yCb) , the luma location $(xB1, yB0)$, the variable $\log2TrafoSize$ set equal to $\log2TrafoSize - 1$, the variable $trafoDepth$ set equal to $trafoDepth + 1$, the variable $nCbS$ and the $(nCbS) \times (nCbS)$ array $resSamples$ as inputs, and the output is a modified version of the $(nCbS) \times (nCbS)$ array $resSamples$.
 4. The decoding process for luma residual blocks as specified in this clause is invoked with the luma location (xCb, yCb) , the luma location $(xB0, yB1)$, the variable $\log2TrafoSize$ set equal to $\log2TrafoSize - 1$, the variable $trafoDepth$ set equal to $trafoDepth + 1$, the variable $nCbS$ and the $(nCbS) \times (nCbS)$ array $resSamples$ as inputs, and the output is a modified version of the $(nCbS) \times (nCbS)$ array $resSamples$.
 5. The decoding process for luma residual blocks as specified in this clause is invoked with the luma location (xCb, yCb) , the luma location $(xB1, yB1)$, the variable $\log2TrafoSize$ set equal to $\log2TrafoSize - 1$, the variable $trafoDepth$ set equal to $trafoDepth + 1$, the variable $nCbS$ and the $(nCbS) \times (nCbS)$ array $resSamples$ as inputs, and the output is a modified version of the $(nCbS) \times (nCbS)$ array $resSamples$.
- Otherwise ($split_transform_flag[xCb + xB0][yCb + yB0][trafoDepth]$ is equal to 0), the following ordered steps apply:
 1. The variable $nTbS$ is set equal to $1 \ll \log2TrafoSize$.
 2. The scaling and transformation process as specified in clause 8.6.2 is invoked with the luma location $(xCb + xB0, yCb + yB0)$, the variable $trafoDepth$, the variable $cIdx$ set equal to 0 and the transform size $trafoSize$ set equal to $nTbS$ as inputs, and the output is an $(nTbS) \times (nTbS)$ array $transformBlock$.
 3. When $explicit_rdpcm_flag[xCb + xB0][yCb + yB0][0]$ is equal to 1, the directional residual modification process for blocks using a transform bypass as specified in clause 8.6.5 is invoked with the variable $mDir$ set equal to $explicit_rdpcm_dir_flag[xCb + xB0][yCb + yB0][0]$, the variable $nTbS$ and the $(nTbS) \times (nTbS)$ array r set equal to the array $transformBlock$ as inputs, and the output is a modified $(nTbS) \times (nTbS)$ array $transformBlock$.
 4. The $(nCbS) \times (nCbS)$ residual sample array of the current coding block $resSamples$ is modified as follows:

$$resSamples[xB0 + i, yB0 + j] = transformBlock[i, j], \text{ with } i = 0..nTbS - 1, j = 0..nTbS - 1 \text{ (8-278)}$$

8.5.4.3 Decoding process for chroma residual blocks

This process is only invoked when $ChromaArrayType$ is not equal to 0.

Inputs to this process are:

- a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a luma location $(xB0, yB0)$ specifying the top-left luma sample of the current chroma block relative to the top-left sample of the current luma coding block,
- a variable $\log2TrafoSize$ specifying the size of the current chroma block in luma samples,
- a variable $trafoDepth$ specifying the hierarchy depth of the current chroma block relative to the chroma coding block,
- a variable $cIdx$ specifying the chroma component of the current block,
- the variables $nCbSw$ and $nCbSh$ specifying the width and height, respectively, of the current chroma coding block,
- an $(nCbSw) \times (nCbSh)$ array $resSamples$ of chroma residual samples.

Output of this process is a modified version of the $(nCbSw) \times (nCbSh)$ array of chroma residual samples.

The variable `splitChromaFlag` is derived as follows:

- If `split_transform_flag[xCb + xB0][yCb + yB0][trafoDepth]` is equal to 1 and one or more of the following conditions are met, `splitChromaFlag` is set equal to 1:
 - `log2TrafoSize` is greater than 3.
 - `ChromaArrayType` is equal to 3.
- Otherwise (`split_transform_flag[xCb + xB0][yCb + yB0][trafoDepth]` is equal to 0 or both `log2TrafoSize` is equal to 3 and `ChromaArrayType` is not equal to 3), `splitChromaFlag` is set equal to 0.

Depending on the value of `splitChromaFlag`, the following applies:

- If `splitChromaFlag` is equal to 1, the following ordered steps apply:
 1. The variables `xB1` and `yB1` are derived as follows:
 - The variable `xB1` is set equal to $xB0 + (1 \ll (\log2TrafoSize - 1))$.
 - The variable `yB1` is set equal to $yB0 + (1 \ll (\log2TrafoSize - 1))$.
 2. The decoding process for residual chroma blocks as specified in this clause is invoked with the luma location (xCb, yCb) , the luma location $(xB0, yB0)$, the variable `log2TrafoSize` set equal to `log2TrafoSize - 1`, the variable `trafoDepth` set equal to `trafoDepth + 1`, the variable `cIdx`, the variable `nCbSw`, the variable `nCbSh` and the $(nCbSw) \times (nCbSh)$ array `resSamples` as inputs, and the output is a modified version of the $(nCbSw) \times (nCbSh)$ array `resSamples`.
 3. The decoding process for residual chroma blocks as specified in this clause is invoked with the luma location (xCb, yCb) , the luma location $(xB1, yB0)$, the variable `log2TrafoSize` set equal to `log2TrafoSize - 1`, the variable `trafoDepth` set equal to `trafoDepth + 1`, the variable `cIdx`, the variable `nCbSw`, the variable `nCbSh` and the $(nCbSw) \times (nCbSh)$ array `resSamples` as inputs, and the output is a modified version of the $(nCbSw) \times (nCbSh)$ array `resSamples`.
 4. The decoding process for residual chroma blocks as specified in this clause is invoked with the luma location (xCb, yCb) , the luma location $(xB0, yB1)$, the variable `log2TrafoSize` set equal to `log2TrafoSize - 1`, the variable `trafoDepth` set equal to `trafoDepth + 1`, the variable `cIdx`, the variable `nCbSw`, the variable `nCbSh` and the $(nCbSw) \times (nCbSh)$ array `resSamples` as inputs, and the output is a modified version of the $(nCbSw) \times (nCbSh)$ array `resSamples`.
 5. The decoding process for residual chroma blocks as specified in this clause is invoked with the luma location (xCb, yCb) , the luma location $(xB1, yB1)$, the variable `log2TrafoSize` set equal to `log2TrafoSize - 1`, the variable `trafoDepth` set equal to `trafoDepth + 1`, the variable `cIdx`, the variable `nCbSw`, the variable `nCbSh` and the $(nCbSw) \times (nCbSh)$ array `resSamples` as inputs, and the output is a modified version of the $(nCbSw) \times (nCbSh)$ array `resSamples`.
- Otherwise (`splitChromaFlag` is equal to 0), for the variable `blkIdx` proceeding over the values $0..(ChromaArrayType == 2 ? 1 : 0)$, the following ordered steps apply:
 1. The variable `nTbS` is set equal to $(1 \ll \log2TrafoSize) / SubWidthC$.
 2. The variable `yBN` is set equal to $yB0 + blkIdx * nTbS * SubHeightC$.
 3. The scaling and transformation process as specified in clause 8.6.2 is invoked with the luma location $(xCb + xB0, yCb + yBN)$, the variable `trafoDepth`, the variable `cIdx` and the transform size `trafoSize` set equal to `nTbS` as inputs, and the output is an $(nTbS) \times (nTbS)$ array `transformBlock`.
 4. When `explicit_rdpem_flag[xCb + xB0][yCb + yBN][cIdx]` is equal to 1, the directional residual modification process for blocks using a transform bypass as specified in clause 8.6.5 is invoked with the variable `mDir` set equal to `explicit_rdpem_dir_flag[xCb + xB0][yCb + yBN][cIdx]`, the variable `nTbS` and the $(nTbS) \times (nTbS)$ array `r` set equal to the array `transformBlock` as inputs, and the output is a modified $(nTbS) \times (nTbS)$ array `transformBlock`.
 5. When `cross_component_prediction_enabled_flag` is equal to 1 and `ChromaArrayType` is equal to 3, the residual modification process for transform blocks using cross-component prediction as specified in clause 8.6.6 is invoked with the transform block location $(xCb + xB0, yCb + yB0)$, the variable `nTbS`, the variable `cIdx`, the $(nTbS) \times (nTbS)$ array `rT` set equal to the corresponding luma residual sample array `transformBlock` of the current transform block and the $(nTbS) \times (nTbS)$ array `r` set equal to the array `transformBlock` as inputs, and the output is a modified $(nTbS) \times (nTbS)$ array `resSamples`.

6. The $(nCbS) \times (nCbS)$ residual sample array of the current coding block $resSamples$ is modified as follows, for $i = 0..nTbS - 1, j = 0..nTbS - 1$:

$$resSamples[(xCb + xB0) / SubWidthC + i, (yCb + yBN) / SubHeightC + j] = transformBlock[i, j] \quad (8-279)$$

8.6 Scaling, transformation and array construction process prior to deblocking filter process

8.6.1 Derivation process for quantization parameters

Input to this process is a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture.

In this process, the variable Qp_Y , the luma quantization parameter Qp'_Y and the chroma quantization parameters Qp'_{Cb} and Qp'_{Cr} are derived.

The luma location (xQg, yQg) , specifies the top-left luma sample of the current quantization group relative to the top-left luma sample of the current picture. The horizontal and vertical positions xQg and yQg are set equal to $xCb - (xCb \& ((1 \ll \text{Log2MinCuQpDeltaSize}) - 1))$ and $yCb - (yCb \& ((1 \ll \text{Log2MinCuQpDeltaSize}) - 1))$, respectively. The luma size of a quantization group, $\text{Log2MinCuQpDeltaSize}$, determines the luma size of the smallest area inside a CTB that shares the same qP_{Y_PREL} .

The predicted luma quantization parameter qP_{Y_PREL} is derived by the following ordered steps:

1. The variable qP_{Y_PREV} is derived as follows:
 - If one or more of the following conditions are true, qP_{Y_PREV} is set equal to SliceQp_Y :
 - The current quantization group is the first quantization group in a slice.
 - The current quantization group is the first quantization group in a tile.
 - The current quantization group is the first quantization group in a CTB row of a tile and `entropy_coding_sync_enabled_flag` is equal to 1.
 - Otherwise, qP_{Y_PREV} is set equal to the luma quantization parameter Qp_Y of the last coding unit in the previous quantization group in decoding order.
2. The availability derivation process for a block in z-scan order as specified in clause 6.4.1 is invoked with the location $(xCurr, yCurr)$ set equal to (xCb, yCb) and the neighbouring location $(xNbY, yNbY)$ set equal to $(xQg - 1, yQg)$ as inputs, and the output is assigned to `availableA`. The variable qP_{Y_A} is derived as follows:
 - If one or more of the following conditions are true, qP_{Y_A} is set equal to qP_{Y_PREV} :
 - `availableA` is equal to FALSE.
 - the CTB address `ctbAddrA` of the CTB containing the luma coding block covering the luma location $(xQg - 1, yQg)$ is not equal to `CtbAddrInTs`, where `ctbAddrA` is derived as follows:

$$\begin{aligned} xTmp &= (xQg - 1) \gg \text{MinTbLog2SizeY} \\ yTmp &= yQg \gg \text{MinTbLog2SizeY} \\ \text{minTbAddrA} &= \text{MinTbAddrZs}[xTmp][yTmp] \\ \text{ctbAddrA} &= \text{minTbAddrA} \gg (2 * (\text{CtbLog2SizeY} - \text{MinTbLog2SizeY})) \end{aligned} \quad (8-280)$$
 - Otherwise, qP_{Y_A} is set equal to the luma quantization parameter Qp_Y of the coding unit containing the luma coding block covering $(xQg - 1, yQg)$.
3. The availability derivation process for a block in z-scan order as specified in clause 6.4.1 is invoked with the location $(xCurr, yCurr)$ set equal to (xCb, yCb) and the neighbouring location $(xNbY, yNbY)$ set equal to $(xQg, yQg - 1)$ as inputs, and the output is assigned to `availableB`. The variable qP_{Y_B} is derived as follows:
 - If one or more of the following conditions are true, qP_{Y_B} is set equal to qP_{Y_PREV} :
 - `availableB` is equal to FALSE.
 - the CTB address `ctbAddrB` of the CTB containing the luma coding block covering the luma location $(xQg, yQg - 1)$ is not equal to `CtbAddrInTs`, where `ctbAddrB` is derived as follows:

$$\begin{aligned} xTmp &= xQg \gg \text{MinTbLog2SizeY} \\ yTmp &= (yQg - 1) \gg \text{MinTbLog2SizeY} \end{aligned}$$

$$\begin{aligned} \text{minTbAddrB} &= \text{MinTbAddrZs}[\text{xTmp}][\text{yTmp}] \\ \text{ctbAddrB} &= \text{minTbAddrB} \gg (2 * (\text{CtbLog2SizeY} - \text{MinTbLog2SizeY})) \end{aligned} \quad (8-281)$$

- Otherwise, qP_{Y_B} is set equal to the luma quantization parameter Qp_Y of the coding unit containing the luma coding block covering ($xQg, yQg - 1$).

4. The predicted luma quantization parameter qP_{Y_PRED} is derived as follows:

$$qP_{Y_PRED} = (qP_{Y_A} + qP_{Y_B} + 1) \gg 1 \quad (8-282)$$

The variable Qp_Y is derived as follows:

$$Qp_Y = ((qP_{Y_PRED} + \text{CuQpDeltaVal} + 52 + 2 * QpBdOffset_Y) \% (52 + QpBdOffset_Y)) - QpBdOffset_Y \quad (8-283)$$

The luma quantization parameter Qp'_Y is derived as follows:

$$Qp'_Y = Qp_Y + QpBdOffset_Y \quad (8-284)$$

When ChromaArrayType is not equal to 0, the following applies:

- The variables qP_{Cb} and qP_{Cr} are derived as follows:
- If $\text{tu_residual_act_flag}[\text{xTbY}][\text{yTbY}]$ is equal to 0, the following applies:

$$qPi_{Cb} = \text{Clip3}(-QpBdOffset_C, 57, Qp_Y + \text{pps_cb_qp_offset} + \text{slice_cb_qp_offset} + \text{CuQpOffset}_{Cb}) \quad (8-285)$$

$$qPi_{Cr} = \text{Clip3}(-QpBdOffset_C, 57, Qp_Y + \text{pps_cr_qp_offset} + \text{slice_cr_qp_offset} + \text{CuQpOffset}_{Cr}) \quad (8-286)$$

- Otherwise ($\text{tu_residual_act_flag}[\text{xTbY}][\text{yTbY}]$ is equal to 1), the following applies:

$$qPi_{Cb} = \text{Clip3}(-QpBdOffset_C, 57, Qp_Y + \text{PpsActQpOffset}_{Cb} + \text{slice_act_cb_qp_offset} + \text{CuQpOffset}_{Cb}) \quad (8-287)$$

$$qPi_{Cr} = \text{Clip3}(-QpBdOffset_C, 57, Qp_Y + \text{PpsActQpOffset}_{Cr} + \text{slice_act_cr_qp_offset} + \text{CuQpOffset}_{Cr}) \quad (8-288)$$

- If ChromaArrayType is equal to 1, the variables qP_{Cb} and qP_{Cr} are set equal to the value of Qp_C as specified in Table 8-10 based on the index qPi equal to qPi_{Cb} and qPi_{Cr} , respectively.
- Otherwise, the variables qP_{Cb} and qP_{Cr} are set equal to $\text{Min}(qPi, 51)$, based on the index qPi equal to qPi_{Cb} and qPi_{Cr} , respectively.
- The chroma quantization parameters for the Cb and Cr components, Qp'_{Cb} and Qp'_{Cr} , are derived as follows:

$$Qp'_{Cb} = qP_{Cb} + QpBdOffset_C \quad (8-289)$$

$$Qp'_{Cr} = qP_{Cr} + QpBdOffset_C \quad (8-290)$$

Table 8-10 – Specification of Qp_C as a function of qPi for ChromaArrayType equal to 1

qPi	< 30	30	31	32	33	34	35	36	37	38	39	40	41	42	43	> 43
Qp_C	= qPi	29	30	31	32	33	33	34	34	35	35	36	36	37	37	= $qPi - 6$

8.6.2 Scaling and transformation process

Inputs to this process are:

- a luma location ($xTbY, yTbY$) specifying the top-left sample of the current luma transform block relative to the top-left luma sample of the current picture,

- a variable `trafoDepth` specifying the hierarchy depth of the current block relative to the coding block,
- a variable `cIdx` specifying the colour component of the current block,
- a variable `nTbS` specifying the size of the current transform block.

Output of this process is the $(nTbS) \times (nTbS)$ array of residual samples r with elements $r[x][y]$.

The quantization parameter qP is derived as follows:

- If `cIdx` is equal to 0, the following applies:

$$qP = \text{Clip3}(0, 51 + QpBdOffsetY, Qp'Y + (tu_residual_act_flag[xTbY][yTbY] ? PpsActQpOffsetY + slice_act_y_qp_offset : 0)) \quad (8-291)$$

- Otherwise, if `cIdx` is equal to 1, the following applies:

$$qP = Qp'_{cb} \quad (8-292)$$

- Otherwise (`cIdx` is equal to 2), the following applies:

$$qP = Qp'_{cr} \quad (8-293)$$

The variables `bitDepth`, `bdShift` and `tsShift` are derived as follows:

$$\text{bitDepth} = (\text{cIdx} == 0) ? \text{BitDepth}_Y : \text{BitDepth}_C \quad (8-294)$$

$$\text{bdShift} = \text{Max}(20 - \text{bitDepth}, \text{extended_precision_processing_flag} ? 11 : 0) \quad (8-295)$$

$$\text{tsShift} = 5 + \text{Log2}(nTbS) \quad (8-296)$$

The variable `rotateCoeffs` is derived as follows:

- If all of the following conditions are true, `rotateCoeffs` is set equal to 1:
 - `transform_skip_rotation_enabled_flag` is equal to 1.
 - `nTbS` is equal to 4.
 - `CuPredMode[xTbY][yTbY]` is equal to `MODE_INTRA`.
- Otherwise, `rotateCoeffs` is set equal to 0.

The $(nTbS) \times (nTbS)$ array of residual samples r is derived as follows:

- If `cu_transquant_bypass_flag` is equal to 1, the following applies:
 - If `rotateCoeffs` is equal to 1, the residual sample array values $r[x][y]$ with $x = 0..nTbS - 1$, $y = 0..nTbS - 1$ are derived as follows:

$$r[x][y] = \text{TransCoeffLevel}[xTbY][yTbY][cIdx][nTbS - x - 1][nTbS - y - 1] \quad (8-297)$$

- Otherwise, the $(nTbS) \times (nTbS)$ array r is set equal to the $(nTbS) \times (nTbS)$ array of transform coefficients `TransCoeffLevel[xTbY][yTbY][cIdx]`.
- Otherwise, the following ordered steps apply:
 1. The scaling process for transform coefficients as specified in clause 8.6.3 is invoked with the transform block location (`xTbY`, `yTbY`), the size of the transform block `nTbS`, the colour component variable `cIdx` and the quantization parameter qP as inputs, and the output is an $(nTbS) \times (nTbS)$ array of scaled transform coefficients d .
 2. The $(nTbS) \times (nTbS)$ array of residual samples r is derived as follows:
 - If `transform_skip_flag[xTbY][yTbY][cIdx]` is equal to 1, the residual sample array values $r[x][y]$ with $x = 0..nTbS - 1$, $y = 0..nTbS - 1$ are derived as follows:

$$r[x][y] = (\text{rotateCoeffs} ? d[nTbS - x - 1][nTbS - y - 1] : d[x][y]) << \text{tsShift} \quad (8-298)$$

- Otherwise (transform_skip_flag[xTbY][yTbY][cIdx] is equal to 0), the transformation process for scaled transform coefficients as specified in clause 8.6.4 is invoked with the transform block location (xTbY, yTbY), the size of the transform block nTbS, the colour component variable cIdx and the (nTbS)x(nTbS) array of scaled transform coefficients d as inputs, and the output is an (nTbS)x(nTbS) array of residual samples r.

3. The residual sample values $r[x][y]$ with $x = 0..nTbS - 1$, $y = 0..nTbS - 1$ are modified as follows:

$$r[x][y] = (r[x][y] + (1 \ll (bdShift - 1))) \gg bdShift \quad (8-299)$$

8.6.3 Scaling process for transform coefficients

Inputs to this process are:

- a luma location (xTbY, yTbY) specifying the top-left sample of the current luma transform block relative to the top-left luma sample of the current picture,
- a variable nTbS specifying the size of the current transform block,
- a variable cIdx specifying the colour component of the current block,
- a variable qP specifying the quantization parameter.

Output of this process is the (nTbS)x(nTbS) array d of scaled transform coefficients with elements $d[x][y]$.

The variables log2TransformRange, bdShift, coeffMin and coeffMax are derived as follows:

- If cIdx is equal to 0, the following applies:

$$\log2TransformRange = \text{extended_precision_processing_flag} ? \text{Max}(15, \text{BitDepth}_Y + 6) : 15 \quad (8-300)$$

$$bdShift = \text{BitDepth}_Y + \text{Log2}(nTbS) + 10 - \log2TransformRange \quad (8-301)$$

$$\text{coeffMin} = \text{CoeffMin}_Y \quad (8-302)$$

$$\text{coeffMax} = \text{CoeffMax}_Y \quad (8-303)$$

- Otherwise, the following applies:

$$\log2TransformRange = \text{extended_precision_processing_flag} ? \text{Max}(15, \text{BitDepth}_C + 6) : 15 \quad (8-304)$$

$$bdShift = \text{BitDepth}_C + \text{Log2}(nTbS) + 10 - \log2TransformRange \quad (8-305)$$

$$\text{coeffMin} = \text{CoeffMin}_C \quad (8-306)$$

$$\text{coeffMax} = \text{CoeffMax}_C \quad (8-307)$$

The list levelScale[] is specified as levelScale[k] = { 40, 45, 51, 57, 64, 72 } with $k = 0..5$.

For the derivation of the scaled transform coefficients $d[x][y]$ with $x = 0..nTbS - 1$, $y = 0..nTbS - 1$, the following applies:

- The scaling factor $m[x][y]$ is derived as follows:
 - If one or more of the following conditions are true, $m[x][y]$ is set equal to 16:
 - scaling_list_enabled_flag is equal to 0.
 - transform_skip_flag[xTbY][yTbY] is equal to 1 and nTbS is greater than 4.
 - Otherwise, the following applies:

$$m[x][y] = \text{ScalingFactor}[\text{sizeId}][\text{matrixId}][x][y] \quad (8-308)$$

Where sizeId is specified in Table 7-3 for the size of the quantization matrix equal to (nTbS)x(nTbS) and matrixId is specified in Table 7-4 for sizeId, CuPredMode[xTbY][yTbY] and cIdx, respectively.

- The scaled transform coefficient $d[x][y]$ is derived as follows:

$$d[x][y] = \text{Clip3}(\text{coeffMin}, \text{coeffMax}, ((\text{TransCoeffLevel}[xTbY][yTbY][cIdx][x][y] * m[x][y] * \text{levelScale}[qP \% 6] << (qP / 6)) + (1 << (bdShift - 1))) >> bdShift) \quad (8-309)$$

8.6.4 Transformation process for scaled transform coefficients

8.6.4.1 General

Inputs to this process are:

- a luma location (xTbY, yTbY) specifying the top-left sample of the current luma transform block relative to the top-left luma sample of the current picture,
- a variable nTbS specifying the size of the current transform block,
- a variable cIdx specifying the colour component of the current block,
- an (nTbS)x(nTbS) array d of scaled transform coefficients with elements d[x][y].

Output of this process is the (nTbS)x(nTbS) array r of residual samples with elements r[x][y].

The variables coeffMin and coeffMax are derived as follows:

- If cIdx is equal to 0, the following applies:

$$\text{coeffMin} = \text{CoeffMin}_Y \quad (8-310)$$

$$\text{coeffMax} = \text{CoeffMax}_Y \quad (8-311)$$

- Otherwise, the following applies:

$$\text{coeffMin} = \text{CoeffMin}_C \quad (8-312)$$

$$\text{coeffMax} = \text{CoeffMax}_C \quad (8-313)$$

Depending on the values of CuPredMode[xTbY][yTbY], nTbS and cIdx, the variable trType is derived as follows:

- If CuPredMode[xTbY][yTbY] is equal to MODE_INTRA, nTbS is equal to 4 and cIdx is equal to 0, trType is set equal to 1.
- Otherwise, trType is set equal to 0.

The (nTbS)x(nTbS) array r of residual samples is derived by the following ordered steps:

1. Each (vertical) column of scaled transform coefficients d[x][y] with x = 0..nTbS – 1, y = 0..nTbS – 1 is transformed to e[x][y] with x = 0..nTbS – 1, y = 0..nTbS – 1 by invoking the one-dimensional transformation process as specified in clause 8.6.4.2 for each column x = 0..nTbS – 1 with the size of the transform block nTbS, the list d[x][y] with y = 0..nTbS – 1 and the transform type variable trType as inputs, and the output is the list e[x][y] with y = 0..nTbS – 1.

2. The intermediate sample values g[x][y] with x = 0..nTbS – 1, y = 0..nTbS – 1 are derived as follows:

$$g[x][y] = \text{Clip3}(\text{coeffMin}, \text{coeffMax}, (e[x][y] + 64) >> 7) \quad (8-314)$$

3. Each (horizontal) row of the resulting array g[x][y] with x = 0..nTbS – 1, y = 0..nTbS – 1 is transformed to r[x][y] with x = 0..nTbS – 1, y = 0..nTbS – 1 by invoking the one-dimensional transformation process as specified in clause 8.6.4.2 for each row y = 0..nTbS – 1 with the size of the transform block nTbS, the list g[x][y] with x = 0..nTbS – 1 and the transform type variable trType as inputs, and the output is the list r[x][y] with x = 0..nTbS – 1.

8.6.4.2 Transformation process

Inputs to this process are:

- a variable nTbS specifying the sample size of scaled transform coefficients,
- a list of scaled transform coefficients x with elements x[j], with j = 0..nTbS – 1.
- a transform type variable trType

Output of this process is the list of transformed samples y with elements $y[i]$, with $i = 0..nTbS - 1$.

Depending on the value of $trType$, the following applies:

- If $trType$ is equal to 1, the following transform matrix multiplication applies:

$$y[i] = \sum_{j=0}^{nTbS-1} transMatrix[i][j] * x[j] \text{ with } i = 0..nTbS - 1 \quad (8-315)$$

where the transform coefficient array $transMatrix$ is specified as follows:

$$transMatrix = \quad (8-316)$$

```
{
{29  55  74  84}
{74  74   0 -74}
{84 -29 -74  55}
{55 -84  74 -29}
}
```

- Otherwise ($trType$ is equal to 0), the following transform matrix multiplication applies:

$$y[i] = \sum_{j=0}^{nTbS-1} transMatrix[i][j * 2^{5-\log_2(nTbS)}] * x[j] \text{ with } i = 0..nTbS - 1, \quad (8-317)$$

where the transform coefficient array $transMatrix$ is specified as follows:

$$transMatrix[m][n] = transMatrixCol0to15[m][n] \text{ with } m = 0..15, n = 0..31 \quad (8-318)$$

$$transMatrixCol0to15 = \quad (8-319)$$

```
{
{64  64  64  64  64  64  64  64  64  64  64  64  64  64  64}
{90  90  88  85  82  78  73  67  61  54  46  38  31  22  13  4}
{90  87  80  70  57  43  25  9  -9 -25 -43 -57 -70 -80 -87 -90}
{90  82  67  46  22  -4 -31 -54 -73 -85 -90 -88 -78 -61 -38 -13}
{89  75  50  18 -18 -50 -75 -89 -89 -75 -50 -18  18  50  75  89}
{88  67  31 -13 -54 -82 -90 -78 -46 -4  38  73  90  85  61  22}
{87  57  9 -43 -80 -90 -70 -25 25  70  90  80  43 -9 -57 -87}
{85  46 -13 -67 -90 -73 -22 38  82  88  54 -4 -61 -90 -78 -31}
{83  36 -36 -83 -83 -36 36  83  83  36 -36 -83 -83 -36 36  83}
{82  22 -54 -90 -61 13  78  85  31 -46 -90 -67  4  73  88  38}
{80  9 -70 -87 -25 57  90  43 -43 -90 -57 25  87  70 -9 -80}
{78 -4 -82 -73 13  85  67 -22 -88 -61 31  90  54 -38 -90 -46}
{75 -18 -89 -50 50  89  18 -75 -75 18  89  50 -50 -89 -18  75}
{73 -31 -90 -22 78  67 -38 -90 -13 82  61 -46 -88 -4  85  54}
{70 -43 -87  9  90  25 -80 -57 57  80 -25 -90 -9  87  43 -70}
{67 -54 -78 38  85 -22 -90  4  90  13 -88 -31  82  46 -73 -61}
{64 -64 -64 64  64 -64 -64 64  64 -64 -64 64  64 -64 -64 64}
{61 -73 -46 82  31 -88 -13 90 -4 -90 22  85 -38 -78 54  67}
{57 -80 -25 90 -9 -87 43  70 -70 -43 87  9 -90 25  80 -57}
{54 -85 -4  88 -46 -61 82  13 -90 38  67 -78 -22 90 -31 -73}
{50 -89 18  75 -75 -18 89 -50 -50 89 -18 -75 75 18 -89 50}
{46 -90 38  54 -90 31  61 -88 22  67 -85 13  73 -82  4  78}
{43 -90 57  25 -87 70  9 -80 80 -9 -70 87 -25 -57 90 -43}
{38 -88 73 -4 -67 90 -46 -31 85 -78 13  61 -90 54  22 -82}
{36 -83 83 -36 -36 83 -83 36 36 -83 83 -36 -36 83 -83 36}
{31 -78 90 -61  4  54 -88 82 -38 -22 73 -90 67 -13 -46 85}
{25 -70 90 -80 43  9 -57 87 -87 57 -9 -43 80 -90 70 -25}
{22 -61 85 -90 73 -38 -4  46 -78 90 -82 54 -13 -31 67 -88}
{18 -50 75 -89 89 -75 50 -18 -18 50 -75 89 -89 75 -50 18}
{13 -38 61 -78 88 -90 85 -73 54 -31  4  22 -46 67 -82 90}
{ 9 -25 43 -57 70 -80 87 -90 90 -87 80 -70 57 -43 25 -9}
{ 4 -13 22 -31 38 -46 54 -61 67 -73 78 -82 85 -88 90 -90}
},
```

$$\text{transMatrix}[m][n] = \text{transMatrixCol16to31}[m-16][n] \text{ with } m = 16..31, n = 0..31, \quad (8-320)$$

$$\text{transMatrixCol16to31} = \quad (8-321)$$

```
{
{ 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 64 }
{ -4 -13 -22 -31 -38 -46 -54 -61 -67 -73 -78 -82 -85 -88 -90 -90 }
{ -90 -87 -80 -70 -57 -43 -25 -9 9 25 43 57 70 80 87 90 }
{ 13 38 61 78 88 90 85 73 54 31 4 -22 -46 -67 -82 -90 }
{ 89 75 50 18 -18 -50 -75 -89 -89 -75 -50 -18 18 50 75 89 }
{ -22 -61 -85 -90 -73 -38 4 46 78 90 82 54 13 -31 -67 -88 }
{ -87 -57 -9 43 80 90 70 25 -25 -70 -90 -80 -43 9 57 87 }
{ 31 78 90 61 4 -54 -88 -82 -38 22 73 90 67 13 -46 -85 }
{ 83 36 -36 -83 -83 -36 36 83 83 36 -36 -83 -83 -36 36 83 }
{ -38 -88 -73 -4 67 90 46 -31 -85 -78 -13 61 90 54 -22 -82 }
{ -80 -9 70 87 25 -57 -90 -43 43 90 57 -25 -87 -70 9 80 }
{ 46 90 38 -54 -90 -31 61 88 22 -67 -85 -13 73 82 4 -78 }
{ 75 -18 -89 -50 50 89 18 -75 -75 18 89 50 -50 -89 -18 75 }
{ -54 -85 4 88 46 -61 -82 13 90 38 -67 -78 22 90 31 -73 }
{ -70 43 87 -9 -90 -25 80 57 -57 -80 25 90 9 -87 -43 70 }
{ 61 73 -46 -82 31 88 -13 -90 -4 90 22 -85 -38 78 54 -67 }
{ 64 -64 -64 64 64 -64 -64 64 64 -64 -64 64 64 -64 64 }
{ -67 -54 78 38 -85 -22 90 4 -90 13 88 -31 -82 46 73 -61 }
{ -57 80 25 -90 9 87 -43 -70 70 43 -87 -9 90 -25 -80 57 }
{ 73 31 -90 22 78 -67 -38 90 -13 -82 61 46 -88 4 85 -54 }
{ 50 -89 18 75 -75 -18 89 -50 -50 89 -18 -75 75 18 -89 50 }
{ -78 -4 82 -73 -13 85 -67 -22 88 -61 -31 90 -54 -38 90 -46 }
{ -43 90 -57 -25 87 -70 -9 80 -80 9 70 -87 25 57 -90 43 }
{ 82 -22 -54 90 -61 -13 78 -85 31 46 -90 67 4 -73 88 -38 }
{ 36 -83 83 -36 -36 83 -83 36 36 -83 83 -36 -36 83 -83 36 }
{ -85 46 13 -67 90 -73 22 38 -82 88 -54 -4 61 -90 78 -31 }
{ -25 70 -90 80 -43 -9 57 -87 87 -57 9 43 -80 90 -70 25 }
{ 88 -67 31 13 -54 82 -90 78 -46 4 38 -73 90 -85 61 -22 }
{ 18 -50 75 -89 89 -75 50 -18 -18 50 -75 89 -89 75 -50 18 }
{ -90 82 -67 46 -22 -4 31 -54 73 -85 90 -88 78 -61 38 -13 }
{ -9 25 -43 57 -70 80 -87 90 -90 87 -80 70 -57 43 -25 9 }
{ 90 -90 88 -85 82 -78 73 -67 61 -54 46 -38 31 -22 13 -4 }
}
```

8.6.5 Residual modification process for blocks using a transform bypass

Inputs to this process are:

- a variable mDir specifying the residual modification direction,
- a variable nTbS specifying the transform block size,
- an (nTbS)x(nTbS) array of residual samples r with elements r[x][y].

Output of this process is the modified (nTbS)x(nTbS) array of residual samples.

Depending upon the value of mDir, the (nTbS)x(nTbS) array of samples r is modified as follows:

- If mDir is equal to 0 (horizontal direction), the array values r[x][y] are modified as follows, for x proceeding over the values 1..nTbS – 1 and y = 0..nTbS – 1:

$$r[x][y] += r[x-1][y] \quad (8-322)$$

- Otherwise (vertical direction), the array values r[x][y] are modified as follows, for y proceeding over the values 1..nTbS – 1 and for x = 0..nTbS – 1:

$$r[x][y] += r[x][y-1] \quad (8-323)$$

8.6.6 Residual modification process for transform blocks using cross-component prediction

This process is only invoked when ChromaArrayType is equal to 3.

Inputs to this process are:

- a luma location (xTbY, yTbY) specifying the top-left sample of the current luma transform block relative to the top-left luma sample of the current picture,
- a variable nTbS specifying the transform block size,

- a variable *cIdx* specifying the colour component of the current block,
- an $(nTbS) \times (nTbS)$ array of luma residual samples r_Y with elements $r_Y[x][y]$,
- an $(nTbS) \times (nTbS)$ array of residual samples r with elements $r[x][y]$.

Output of this process is the modified $(nTbS) \times (nTbS)$ array r of residual samples.

The $(nTbS) \times (nTbS)$ array of residual samples r with $x = 0..nTbS - 1$, $y = 0..nTbS - 1$ is modified as follows:

$$r[x][y] += (ResScaleVal[cIdx][xTbY][yTbY] * ((r_Y[x][y] << BitDepth_C) >> BitDepth_Y)) >> 3 \quad (8-324)$$

It is a requirement of bitstream conformance that the luma residual samples $r_Y[x][y]$ shall be in the range of *CoeffMinY* to *CoeffMaxY*, inclusive, and the input residual samples $r[x][y]$ shall be in the range of *CoeffMinC* to *CoeffMaxC*, inclusive.

8.6.7 Picture construction process prior to in-loop filter process

Inputs to this process are:

- a location $(xCurr, yCurr)$ specifying the top-left sample of the current block relative to the top-left sample of the current picture component,
- the variables *nCurrSw* and *nCurrSh* specifying the width and height, respectively, of the current block,
- a variable *cIdx* specifying the colour component of the current block,
- an $(nCurrSw) \times (nCurrSh)$ array *predSamples* specifying the predicted samples of the current block,
- an $(nCurrSw) \times (nCurrSh)$ array *resSamples* specifying the residual samples of the current block.

Depending on the value of the colour component *cIdx*, the following assignments are made:

- If *cIdx* is equal to 0, *recSamples* corresponds to the reconstructed picture sample array S_L and the function *clipCidx1* corresponds to *Clip1Y*.
- Otherwise, if *cIdx* is equal to 1, *recSamples* corresponds to the reconstructed chroma sample array S_{Cb} and the function *clipCidx1* corresponds to *Clip1C*.
- Otherwise (*cIdx* is equal to 2), *recSamples* corresponds to the reconstructed chroma sample array S_{Cr} and the function *clipCidx1* corresponds to *Clip1C*.

The $(nCurrSw) \times (nCurrSh)$ block of the reconstructed sample array *recSamples* at location $(xCurr, yCurr)$ is derived as follows:

$$recSamples[xCurr + i][yCurr + j] = clipCidx1(predSamples[i][j] + resSamples[i][j]) \quad (8-325)$$

with $i = 0..nCurrSw - 1$, $j = 0..nCurrSh - 1$

8.6.8 Residual modification process for blocks using adaptive colour transform

This process is only invoked when *ChromaArrayType* is equal to 3.

8.6.8.1 General

Inputs to this process are:

- a sample location $(xTb0, yTb0)$ specifying the top-left sample of the current block relative to the top left sample of the current picture,
- a variable *log2TrafoSize* specifying the size of the current block,
- a variable *trafoDepth* specifying the hierarchy depth of the current block relative to the coding unit,
- an array *resSampleArrayL* specifying the luma residual samples of the current block,
- an array *resSampleArrayCb* specifying the Cb residual samples of the current block,
- an array *resSampleArrayCr* specifying the Cr residual samples of the current block.

Outputs of this process are the modified arrays `resSampleY`, `resSampleCb` and `resSampleCr` of residual samples of the current block.

Depending on the value of `split_transform_flag[xTb0][yTb0][trafoDepth]`, the following applies:

- If `split_transform_flag[xTb0][yTb0][trafoDepth]` is equal to 1, the following ordered steps apply:
 1. The variables `xTb1` and `yTb1` are derived as follows:
 - The variable `xTb1` is set equal to $xTb0 + (1 \ll (\log_2TrafoSize - 1))$.
 - The variable `yTb1` is set equal to $yTb0 + (1 \ll (\log_2TrafoSize - 1))$.
 2. The residual modification process for blocks using adaptive colour transform as specified in this clause is invoked with the location (`xTb0`, `yTb0`), the variable `log2TrafoSize` set equal to `log2TrafoSize - 1`, the variable `trafoDepth` set equal to `trafoDepth + 1`, the residual sample arrays `resSampeArrayL`, `resSampleArrayCb` and `resSampleArrayCr` as inputs and the output are modified versions of `resSampeArrayL`, `resSampleArrayCb` and `resSampleArrayCr`.
 3. The residual modification process for blocks using adaptive colour transform as specified in this clause is invoked with the location (`xTb1`, `yTb0`), the variable `log2TrafoSize` set equal to `log2TrafoSize - 1`, the variable `trafoDepth` set equal to `trafoDepth + 1`, the residual sample arrays `resSampeArrayL`, `resSampleArrayCb` and `resSampleArrayCr` as inputs and the output are modified versions of `resSampeArrayL`, `resSampleArrayCb` and `resSampleArrayCr`.
 4. The residual modification process for blocks using adaptive colour transform as specified in this clause is invoked with the location (`xTb0`, `yTb1`), the variable `log2TrafoSize` set equal to `log2TrafoSize - 1`, the variable `trafoDepth` set equal to `trafoDepth + 1`, the residual sample arrays `resSampeArrayL`, `resSampleArrayCb` and `resSampleArrayCr` as inputs and the output are modified versions of `resSampeArrayL`, `resSampleArrayCb` and `resSampleArrayCr`.
 5. The residual modification process for blocks using adaptive colour transform as specified in this clause is invoked with the location (`xTb1`, `yTb1`), the variable `log2TrafoSize` set equal to `log2TrafoSize - 1`, the variable `trafoDepth` set equal to `trafoDepth + 1`, the residual sample arrays `resSampeArrayL`, `resSampleArrayCb` and `resSampleArrayCr` as inputs and the output are modified versions of `resSampeArrayL`, `resSampleArrayCb` and `resSampleArrayCr`.
- Otherwise (`split_transform_flag[xTb0][yTb0][trafoDepth]` is equal to 0), the following ordered steps apply:
 1. The variable `nTbS` specifying the current transform block is set equal to $(1 \ll \log_2TrafoSize)$.
 2. The variable `nCbS` specifying the size of the current coding block is derived as $nCbS = 1 \ll (\log_2TrafoSize + trafoDepth)$.
 3. The sample location (`xTbInCb`, `yTbInCb`) specifying the top-left sample of the current transform block relative to the top-left sample of the current coding block are derived as $xTbInCb = xTb0 \& (nCbS - 1)$ and $yTbInCb = yTb0 \& (nCbS - 1)$.

When `tu_residual_act_flag[xTb0][yTb0]` is equal to 1, the adaptive colour transformation process as specified in clause 8.6.8.2 is invoked with the sample location (`xTb0`, `yTb0`) set to (`xTbInCb`, `yTbInCb`), the variable `blkSize` set equal to `nTbS`, the array `rY` set equal to `resSampleArrayL`, the array `rCb` set equal to `resSampleArrayCb`, and the array `rCr` set equal to `resSampleArrayCr` as inputs, and the outputs are modified versions of the three residual sample arrays.

8.6.8.2 Adaptive colour transformation process

Inputs to this process are:

- a sample location (`xTb0`, `yTb0`) specifying the top-left sample of the current transform block relative to the top left sample of the current coding block,
- a variable `blkSize` specifying the block size of the transform block to be modified,
- an array of luma residual samples `rY` of the current coding block,
- an array of chroma residual samples `rCb` of the current coding block,
- an array of chroma residual samples `rCr` of the current coding block.

Outputs of this process are:

- a modified array `rY` of luma residual samples,

- a modified array rCb of chroma residual samples,
- a modified array rCr of chroma residual samples.

The arrays of residual samples rY, rCb and rCr with $x = x_{Tb0}..x_{Tb0} + blkSize - 1$, $y = y_{Tb0}..y_{Tb0} + blkSize - 1$ are modified as follows:

$$rY[x][y] = Clip3(CoeffMinY, CoeffMaxY, rY[x][y]) \quad (8-326)$$

$$rCb[x][y] = Clip3(CoeffMinC, CoeffMaxC, rCb[x][y]) \quad (8-327)$$

$$rCr[x][y] = Clip3(CoeffMinC, CoeffMaxC, rCr[x][y]) \quad (8-328)$$

- When cu_transquant_bypass_flag is equal to 0, the following ordered steps apply:

1. The variables deltaBDY and deltaBDC are derived as follows:

$$deltaBDY = Max(BitDepth_Y, BitDepth_C) - BitDepth_Y \quad (8-329)$$

$$deltaBDC = Max(BitDepth_Y, BitDepth_C) - BitDepth_C \quad (8-330)$$

$$offsetBDY = deltaBDY ? 1 << (deltaBDY - 1) : 0 \quad (8-331)$$

$$offsetBDC = deltaBDC ? 1 << (deltaBDC - 1) : 0 \quad (8-332)$$

2. Residual samples rY[x][y], rCb[x][y] and rCr[x][y] are modified as follows:

$$rY[x][y] = rY[x][y] << deltaBDY \quad (8-333)$$

$$rCb[x][y] = rCb[x][y] << (deltaBDC + 1) \quad (8-334)$$

$$rCr[x][y] = rCr[x][y] << (deltaBDC + 1) \quad (8-335)$$

- Residual samples rY[x][y], rCb[x][y] and rCr[x][y] are modified as follows:

$$tmp = rY[x][y] - (rCb[x][y] >> 1) \quad (8-336)$$

$$rY[x][y] = tmp + rCb[x][y] \quad (8-337)$$

$$rCb[x][y] = tmp - (rCr[x][y] >> 1) \quad (8-338)$$

$$rCr[x][y] += rCb[x][y] \quad (8-339)$$

- When cu_transquant_bypass_flag is equal to 0, the following applies:

$$rY[x][y] = (rY[x][y] + offsetBDY) >> deltaBDY \quad (8-340)$$

$$rCb[x][y] = (rCb[x][y] + offsetBDC) >> deltaBDC \quad (8-341)$$

$$rCr[x][y] = (rCr[x][y] + offsetBDC) >> deltaBDC \quad (8-342)$$

8.7 In-loop filter process

8.7.1 General

This clause specifies the application of two in-loop filters. When the in-loop filter process is specified as optional in Annex A, the application of either or both of these filters is optional.

The two in-loop filters, namely deblocking filter and sample adaptive offset filter, are applied as specified by the following ordered steps:

1. For the deblocking filter, the following applies:

- The deblocking filter process as specified in clause 8.7.2 is invoked with the reconstructed picture sample array S_L and, when ChromaArrayType is not equal to 0, the arrays S_{Cb} and S_{Cr} as inputs, and the modified reconstructed picture sample array S'_L and, when ChromaArrayType is not equal to 0, the arrays S'_{Cb} and S'_{Cr} after deblocking as outputs.
 - The array S'_L and, when ChromaArrayType is not equal to 0, the arrays S'_{Cb} and S'_{Cr} are assigned to the array S_L and, when ChromaArrayType is not equal to 0, the arrays S_{Cb} and S_{Cr} (which represent the decoded picture), respectively.
2. When sample_adaptive_offset_enabled_flag is equal to 1, the following applies:
- The sample adaptive offset process as specified in clause 8.7.3 is invoked with the reconstructed picture sample array S_L and, when ChromaArrayType is not equal to 0, the arrays S_{Cb} and S_{Cr} as inputs, and the modified reconstructed picture sample array S'_L and, when ChromaArrayType is not equal to 0, the arrays S'_{Cb} and S'_{Cr} after sample adaptive offset as outputs.
 - The array S'_L and, when ChromaArrayType is not equal to 0, the arrays S'_{Cb} and S'_{Cr} are assigned to the array S_L and, when ChromaArrayType is not equal to 0, the arrays S_{Cb} and S_{Cr} (which represent the decoded picture), respectively.

8.7.2 Deblocking filter process

8.7.2.1 General

Inputs to this process are the reconstructed picture prior to deblocking, i.e., the array $recPicture_L$ and, when ChromaArrayType is not equal to 0, the arrays $recPicture_{Cb}$ and $recPicture_{Cr}$.

Outputs of this process are the modified reconstructed picture after deblocking, i.e., the array $recPicture_L$ and, when ChromaArrayType is not equal to 0, the arrays $recPicture_{Cb}$ and $recPicture_{Cr}$.

The vertical edges in a picture are filtered first. Then the horizontal edges in a picture are filtered with samples modified by the vertical edge filtering process as input. The vertical and horizontal edges in the CTBs of each CTU are processed separately on a coding unit basis. The vertical edges of the coding blocks in a coding unit are filtered starting with the edge on the left-hand side of the coding blocks proceeding through the edges towards the right-hand side of the coding blocks in their geometrical order. The horizontal edges of the coding blocks in a coding unit are filtered starting with the edge on the top of the coding blocks proceeding through the edges towards the bottom of the coding blocks in their geometrical order.

NOTE – Although the filtering process is specified on a picture basis in this Specification, the filtering process can be implemented on a coding unit basis with an equivalent result, provided the decoder properly accounts for the processing dependency order so as to produce the same output values.

The deblocking filter process is applied to all prediction block edges and transform block edges of a picture, except the following types of edges:

- Edges that are at the boundary of the picture,
- Edges that coincide with tile boundaries when loop_filter_across_tiles_enabled_flag is equal to 0,
- Edges that coincide with upper or left boundaries of slices with slice_loop_filter_across_slices_enabled_flag equal to 0 or slice_deblocking_filter_disabled_flag equal to 1,
- Edges within slices with slice_deblocking_filter_disabled_flag equal to 1,
- Edges that do not correspond to 8x8 sample grid boundaries of the considered component,
- Edges within chroma components for which both sides of the edge use inter prediction,
- Edges of chroma transform blocks that are not edges of the associated transform unit.

The edge type, vertical or horizontal, is represented by the variable edgeType as specified in Table 8-11.

Table 8-11 – Name of association to edgeType

edgeType	Name of edgeType
0 (vertical edge)	EDGE_VER
1 (horizontal edge)	EDGE_HOR

When slice_deblocking_filter_disabled_flag of the current slice is equal to 0, for each coding unit with luma coding block size log2CbSize and location of top-left sample of the luma coding block (x_{Cb} , y_{Cb}), the vertical edges are filtered by the following ordered steps:

1. The luma coding block size $nCbS$ is set equal to $1 \ll \log_2 CbSize$.
2. The variable `filterLeftCbEdgeFlag` is derived as follows:
 - If one or more of the following conditions are true, `filterLeftCbEdgeFlag` is set equal to 0:
 - The left boundary of the current luma coding block is the left boundary of the picture.
 - The left boundary of the current luma coding block is the left boundary of the tile and `loop_filter_across_tiles_enabled_flag` is equal to 0.
 - The left boundary of the current luma coding block is the left boundary of the slice and `slice_loop_filter_across_slices_enabled_flag` is equal to 0.
 - Otherwise, `filterLeftCbEdgeFlag` is set equal to 1.
3. All elements of the two-dimensional $(nCbS) \times (nCbS)$ array `verEdgeFlags` are initialized to be equal to zero.
4. The derivation process of transform block boundary specified in clause 8.7.2.2 is invoked with the luma location (xCb, yCb) , the luma location $(xB0, yB0)$ set equal to $(0, 0)$, the transform block size $\log_2 TrafoSize$ set equal to $\log_2 CbSize$, the variable `trafoDepth` set equal to 0, the variable `filterLeftCbEdgeFlag`, the array `verEdgeFlags` and the variable `edgeType` set equal to `EDGE_VER` as inputs, and the modified array `verEdgeFlags` as output.
5. The derivation process of prediction block boundary specified in clause 8.7.2.3 is invoked with the luma coding block size $\log_2 CbSize$, the prediction partition mode `PartMode`, the array `verEdgeFlags` and the variable `edgeType` set equal to `EDGE_VER` as inputs, and the modified array `verEdgeFlags` as output.
6. The derivation process of the boundary filtering strength specified in clause 8.7.2.4 is invoked with the reconstructed luma picture sample array prior to deblocking `recPictureL`, the luma location (xCb, yCb) , the luma coding block size $\log_2 CbSize$, the variable `edgeType` set equal to `EDGE_VER` and the array `verEdgeFlags` as inputs, and an $(nCbS) \times (nCbS)$ array `verBs` as output.
7. The vertical edge filtering process for a coding unit as specified in clause 8.7.2.5.1 is invoked with the reconstructed picture prior to deblocking, i.e., the array `recPictureL` and, when `ChromaArrayType` is not equal to 0, the arrays `recPictureCb` and `recPictureCr`, the luma location (xCb, yCb) , the luma coding block size $\log_2 CbSize$ and the array `verBs` as inputs, and the modified reconstructed picture, i.e., the array `recPictureL` and, when `ChromaArrayType` is not equal to 0, the arrays `recPictureCb` and `recPictureCr`, as output.

When `slice_deblocking_filter_disabled_flag` of the current slice is equal to 0, for each coding unit with luma coding block size $\log_2 CbSize$ and location of top-left sample of the luma coding block (xCb, yCb) , the horizontal edges are filtered by the following ordered steps:

1. The luma coding block size $nCbS$ is set equal to $1 \ll \log_2 CbSize$.
2. The variable `filterTopCbEdgeFlag` is derived as follows:
 - If one or more of the following conditions are true, the variable `filterTopCbEdgeFlag` is set equal to 0:
 - The top boundary of the current luma coding block is the top boundary of the picture.
 - The top boundary of the current luma coding block is the top boundary of the tile and `loop_filter_across_tiles_enabled_flag` is equal to 0.
 - The top boundary of the current luma coding block is the top boundary of the slice and `slice_loop_filter_across_slices_enabled_flag` is equal to 0.
 - Otherwise, the variable `filterTopCbEdgeFlag` is set equal to 1.
3. All elements of the two-dimensional $(nCbS) \times (nCbS)$ array `horEdgeFlags` are initialized to zero.
4. The derivation process of transform block boundary specified in clause 8.7.2.2 is invoked with the luma location (xCb, yCb) , the luma location $(xB0, yB0)$ set equal to $(0, 0)$, the transform block size $\log_2 TrafoSize$ set equal to $\log_2 CbSize$, the variable `trafoDepth` set equal to 0, the variable `filterTopCbEdgeFlag`, the array `horEdgeFlags` and the variable `edgeType` set equal to `EDGE_HOR` as inputs, and the modified array `horEdgeFlags` as output.
5. The derivation process of prediction block boundary specified in clause 8.7.2.3 is invoked with the luma coding block size $\log_2 CbSize$, the prediction partition mode `PartMode`, the array `horEdgeFlags` and the variable `edgeType` set equal to `EDGE_HOR` as inputs, and the modified array `horEdgeFlags` as output.
6. The derivation process of the boundary filtering strength specified in clause 8.7.2.4 is invoked with the reconstructed luma picture sample array prior to deblocking `recPictureL`, the luma location (xCb, yCb) , the luma coding block size $\log_2 CbSize$, the variable `edgeType` set equal to `EDGE_HOR` and the array `horEdgeFlags` as inputs, and an $(nCbS) \times (nCbS)$ array `horBs` as output.

7. The horizontal edge filtering process for a coding unit as specified in clause 8.7.2.5.2 is invoked with the modified reconstructed picture, i.e., the array recPicture_L and, when ChromaArrayType is not equal to 0, the arrays recPicture_{Cb} and recPicture_{Cr} , the luma location (x_{Cb}, y_{Cb}) , the luma coding block size $\log_2\text{CbSize}$, and the array horBs as inputs and the modified reconstructed picture, i.e., the array recPicture_L and, when ChromaArrayType is not equal to 0, the arrays recPicture_{Cb} and recPicture_{Cr} , as output.

8.7.2.2 Derivation process of transform block boundary

Inputs to this process are:

- a luma location (x_{Cb}, y_{Cb}) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a luma location (x_{B0}, y_{B0}) specifying the top-left sample of the current luma block relative to the top-left sample of the current luma coding block,
- a variable $\log_2\text{TrafoSize}$ specifying the size of the current block,
- a variable trafoDepth ,
- a variable filterEdgeFlag ,
- a two-dimensional $(n_{CbS}) \times (n_{CbS})$ array edgeFlags ,
- a variable edgeType specifying whether a vertical (EDGE_VER) or a horizontal (EDGE_HOR) edge is filtered.

Output of this process is the modified two-dimensional $(n_{CbS}) \times (n_{CbS})$ array edgeFlags .

Depending on the value of $\text{split_transform_flag}[x_{Cb} + x_{B0}][y_{Cb} + y_{B0}][\text{trafoDepth}]$, the following applies:

- If $\text{split_transform_flag}[x_{Cb} + x_{B0}][y_{Cb} + y_{B0}][\text{trafoDepth}]$ is equal to 1, the following ordered steps apply:
 1. The variables x_{B1} and y_{B1} are derived as follows:
 - The variable x_{B1} is set equal to $x_{B0} + (1 \ll (\log_2\text{TrafoSize} - 1))$.
 - The variable y_{B1} is set equal to $y_{B0} + (1 \ll (\log_2\text{TrafoSize} - 1))$.
 2. The derivation process of transform block boundary as specified in this clause is invoked with the luma location (x_{Cb}, y_{Cb}) , the luma location (x_{B0}, y_{B0}) , the variable $\log_2\text{TrafoSize}$ set equal to $\log_2\text{TrafoSize} - 1$, the variable trafoDepth set equal to $\text{trafoDepth} + 1$, the variable filterEdgeFlag , the array edgeFlags and the variable edgeType as inputs, and the output is the modified version of array edgeFlags .
 3. The derivation process of transform block boundary as specified in this clause is invoked with the luma location (x_{Cb}, y_{Cb}) , the luma location (x_{B1}, y_{B0}) , the variable $\log_2\text{TrafoSize}$ set equal to $\log_2\text{TrafoSize} - 1$, the variable trafoDepth set equal to $\text{trafoDepth} + 1$, the variable filterEdgeFlag , the array edgeFlags and the variable edgeType as inputs, and the output is the modified version of array edgeFlags .
 4. The derivation process of transform block boundary as specified in this clause is invoked with the luma location (x_{Cb}, y_{Cb}) , the luma location (x_{B0}, y_{B1}) , the variable $\log_2\text{TrafoSize}$ set equal to $\log_2\text{TrafoSize} - 1$, the variable trafoDepth set equal to $\text{trafoDepth} + 1$, the variable filterEdgeFlag , the array edgeFlags and the variable edgeType as inputs, and the output is the modified version of array edgeFlags .
 5. The derivation process of transform block boundary as specified in this clause is invoked with the luma location (x_{Cb}, y_{Cb}) , the luma location (x_{B1}, y_{B1}) , the variable $\log_2\text{TrafoSize}$ set equal to $\log_2\text{TrafoSize} - 1$, the variable trafoDepth set equal to $\text{trafoDepth} + 1$, the variable filterEdgeFlag , the array edgeFlags and the variable edgeType as inputs, and the output is the modified version of array edgeFlags .
- Otherwise ($\text{split_transform_flag}[x_{Cb} + x_{B0}][y_{Cb} + y_{B0}][\text{trafoDepth}]$ is equal to 0), the following applies:
 - If edgeType is equal to EDGE_VER , the value of $\text{edgeFlags}[x_{B0}][y_{B0} + k]$ for $k = 0..(1 \ll \log_2\text{TrafoSize}) - 1$ is derived as follows:
 - If x_{B0} is equal to 0, $\text{edgeFlags}[x_{B0}][y_{B0} + k]$ is set equal to filterEdgeFlag .
 - Otherwise, $\text{edgeFlags}[x_{B0}][y_{B0} + k]$ is set equal to 1.
 - Otherwise (edgeType is equal to EDGE_HOR), the value of $\text{edgeFlags}[x_{B0} + k][y_{B0}]$ for $k = 0..(1 \ll \log_2\text{TrafoSize}) - 1$ is derived as follows:
 - If y_{B0} is equal to 0, $\text{edgeFlags}[x_{B0} + k][y_{B0}]$ is set equal to filterEdgeFlag .
 - Otherwise, $\text{edgeFlags}[x_{B0} + k][y_{B0}]$ is set equal to 1.

8.7.2.3 Derivation process of prediction block boundary

Inputs to this process are:

- a variable $\log_2\text{CbSize}$ specifying the luma coding block size,
- a prediction partition mode PartMode ,
- a two-dimensional $(n\text{CbS}) \times (n\text{CbS})$ array edgeFlags ,
- a variable edgeType specifying whether a vertical (EDGE_VER) or a horizontal (EDGE_HOR) edge is filtered.

Output of this process is the modified two-dimensional $(n\text{CbS}) \times (n\text{CbS})$ array edgeFlags .

Depending on the values of edgeType and PartMode , the following applies for $k = 0..(1 \ll \log_2\text{CbSize}) - 1$:

- If edgeType is equal to EDGE_VER , the following applies:
 - When PartMode is equal to PART_Nx2N or PART_NxN , $\text{edgeFlags}[1 \ll (\log_2\text{CbSize} - 1)][k]$ is set equal to 1.
 - When PartMode is equal to PART_nLx2N , $\text{edgeFlags}[1 \ll (\log_2\text{CbSize} - 2)][k]$ is set equal to 1.
 - When PartMode is equal to PART_nRx2N , $\text{edgeFlags}[3 * (1 \ll (\log_2\text{CbSize} - 2))][k]$ is set equal to 1.
- Otherwise (edgeType is equal to EDGE_HOR), the following applies:
 - When PartMode is equal to PART_2NxN or PART_NxN , $\text{edgeFlags}[k][1 \ll (\log_2\text{CbSize} - 1)]$ is set equal to 1.
 - When PartMode is equal to PART_2NxN_U , $\text{edgeFlags}[k][1 \ll (\log_2\text{CbSize} - 2)]$ is set equal to 1.
 - When PartMode is equal to PART_2NxN_D , $\text{edgeFlags}[k][3 * (1 \ll (\log_2\text{CbSize} - 2))]$ is set equal to 1.

8.7.2.4 Derivation process of boundary filtering strength

Inputs to this process are:

- a luma picture sample array recPicture_L ,
- a luma location $(x\text{Cb}, y\text{Cb})$ specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a variable $\log_2\text{CbSize}$ specifying the size of the current luma coding block,
- a variable edgeType specifying whether a vertical (EDGE_VER) or a horizontal (EDGE_HOR) edge is filtered,
- a two-dimensional $(n\text{CbS}) \times (n\text{CbS})$ array edgeFlags .

Output of this process is a two-dimensional $(n\text{CbS}) \times (n\text{CbS})$ array $b\text{S}$ specifying the boundary filtering strength.

The variables $x\text{D}_i$, $y\text{D}_j$, $x\text{N}$ and $y\text{N}$ are derived as follows:

- If edgeType is equal to EDGE_VER , $x\text{D}_i$ is set equal to $(i \ll 3)$, $y\text{D}_j$ is set equal to $(j \ll 2)$, $x\text{N}$ is set equal to $(1 \ll (\log_2\text{CbSize} - 3)) - 1$ and $y\text{N}$ is set equal to $(1 \ll (\log_2\text{CbSize} - 2)) - 1$.
- Otherwise (edgeType is equal to EDGE_HOR), $x\text{D}_i$ is set equal to $(i \ll 2)$, $y\text{D}_j$ is set equal to $(j \ll 3)$, $x\text{N}$ is set equal to $(1 \ll (\log_2\text{CbSize} - 2)) - 1$ and $y\text{N}$ is set equal to $(1 \ll (\log_2\text{CbSize} - 3)) - 1$.

For $x\text{D}_i$ with $i = 0..x\text{N}$ and $y\text{D}_j$ with $j = 0..y\text{N}$, the following applies:

- If $\text{edgeFlags}[x\text{D}_i][y\text{D}_j]$ is equal to 0, the variable $b\text{S}[x\text{D}_i][y\text{D}_j]$ is set equal to 0.
- Otherwise ($\text{edgeFlags}[x\text{D}_i][y\text{D}_j]$ is equal to 1), the following applies:
 - The sample values p_0 and q_0 are derived as follows:
 - If edgeType is equal to EDGE_VER , p_0 is set equal to $\text{recPicture}_L[x\text{Cb} + x\text{D}_i - 1][y\text{Cb} + y\text{D}_j]$ and q_0 is set equal to $\text{recPicture}_L[x\text{Cb} + x\text{D}_i][y\text{Cb} + y\text{D}_j]$.
 - Otherwise (edgeType is equal to EDGE_HOR), p_0 is set equal to $\text{recPicture}_L[x\text{Cb} + x\text{D}_i][y\text{Cb} + y\text{D}_j - 1]$ and q_0 is set equal to $\text{recPicture}_L[x\text{Cb} + x\text{D}_i][y\text{Cb} + y\text{D}_j]$.
 - The variable $b\text{S}[x\text{D}_i][y\text{D}_j]$ is derived as follows:
 - If the sample p_0 or q_0 is in the luma coding block of a coding unit coded with intra prediction mode, $b\text{S}[x\text{D}_i][y\text{D}_j]$ is set equal to 2.

- Otherwise, if the block edge is also a transform block edge and the sample p_0 or q_0 is in a luma transform block which contains one or more non-zero transform coefficient levels, $bS[xD_i][yD_j]$ is set equal to 1.
- Otherwise, if one or more of the following conditions are true, $bS[xD_i][yD_j]$ is set equal to 1:
 - For the prediction of the luma prediction block containing the sample p_0 different reference pictures or a different number of motion vectors are used than for the prediction of the luma prediction block containing the sample q_0 .

NOTE 1 – The determination of whether the reference pictures used for the two luma prediction blocks are the same or different is based only on which pictures are referenced, without regard to whether a prediction is formed using an index into reference picture list 0 or an index into reference picture list 1, and also without regard to whether the index position within a reference picture list is different.

NOTE 2 – The number of motion vectors that are used for the prediction of a luma prediction block with top-left luma sample covering (xPb, yPb) , is equal to $PredFlagL0[xPb][yPb] + PredFlagL1[xPb][yPb]$.
 - One motion vector is used to predict the luma prediction block containing the sample p_0 and one motion vector is used to predict the luma prediction block containing the sample q_0 , and the absolute difference between the horizontal or vertical component of the motion vectors used is greater than or equal to 4 in units of quarter luma samples.
 - Two motion vectors and two different reference pictures are used to predict the luma prediction block containing the sample p_0 , two motion vectors for the same two reference pictures are used to predict the luma prediction block containing the sample q_0 and the absolute difference between the horizontal or vertical component of the two motion vectors used in the prediction of the two luma prediction blocks for the same reference picture is greater than or equal to 4 in units of quarter luma samples.
 - Two motion vectors for the same reference picture are used to predict the luma prediction block containing the sample p_0 , two motion vectors for the same reference picture are used to predict the luma prediction block containing the sample q_0 and both of the following conditions are true:
 - The absolute difference between the horizontal or vertical component of list 0 motion vectors used in the prediction of the two luma prediction blocks is greater than or equal to 4 in quarter luma samples, or the absolute difference between the horizontal or vertical component of the list 1 motion vectors used in the prediction of the two luma prediction blocks is greater than or equal to 4 in units of quarter luma samples.
 - The absolute difference between the horizontal or vertical component of list 0 motion vector used in the prediction of the luma prediction block containing the sample p_0 and the list 1 motion vector used in the prediction of the luma prediction block containing the sample q_0 is greater than or equal to 4 in units of quarter luma samples, or the absolute difference between the horizontal or vertical component of the list 1 motion vector used in the prediction of the luma prediction block containing the sample p_0 and list 0 motion vector used in the prediction of the luma prediction block containing the sample q_0 is greater than or equal to 4 in units of quarter luma samples.
- Otherwise, the variable $bS[xD_i][yD_j]$ is set equal to 0.

8.7.2.5 Edge filtering process

8.7.2.5.1 Vertical edge filtering process

Inputs to this process are:

- the picture sample array $recPicture_L$ and, when $ChromaArrayType$ is not equal to 0, the arrays $recPicture_{Cb}$ and $recPicture_{Cr}$,
- a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a variable $log2CbSize$ specifying the size of the current luma coding block,
- an array bS specifying the boundary filtering strength.

Outputs of this process are the modified picture sample array $recPicture_L$ and, when $ChromaArrayType$ is not equal to 0, the arrays $recPicture_{Cb}$ and $recPicture_{Cr}$.

The filtering process for edges in the luma coding block of the current coding unit consists of the following ordered steps:

1. The variable nD is set equal to $1 \ll (\log_2 CbSize - 3)$.
2. For xD_k equal to $k \ll 3$ with $k = 0..nD - 1$ and yD_m equal to $m \ll 2$ with $m = 0..nD * 2 - 1$, the following applies:
 - When $bS[xD_k][yD_m]$ is greater than 0, the following ordered steps apply:
 - a. The decision process for luma block edges as specified in clause 8.7.2.5.3 is invoked with the luma picture sample array $recPicture_L$, the location of the luma coding block (xCb, yCb) , the luma location of the block (xD_k, yD_m) , a variable $edgeType$ set equal to $EDGE_VER$ and the boundary filtering strength $bS[xD_k][yD_m]$ as inputs, and the decisions dE , dEp and dEq , and the variables β and t_c as outputs.
 - b. The filtering process for luma block edges as specified in clause 8.7.2.5.4 is invoked with the luma picture sample array $recPicture_L$, the location of the luma coding block (xCb, yCb) , the luma location of the block (xD_k, yD_m) , a variable $edgeType$ set equal to $EDGE_VER$, the decisions dE , dEp and dEq , and the variables β and t_c as inputs, and the modified luma picture sample array $recPicture_L$ as output.

When $ChromaArrayType$ is not equal to 0, the following applies.

The filtering process for edges in the chroma coding blocks of current coding unit consists of the following ordered steps:

1. The variable nD is set equal to $1 \ll (\log_2 CbSize - 3)$.
2. The variable $edgeSpacing$ is set equal to $8 / SubWidthC$.
3. The variable $edgeSections$ is set equal to $nD * (2 / SubHeightC)$.
4. For xD_k equal to $k * edgeSpacing$ with $k = 0..nD - 1$ and yD_m equal to $m \ll 2$ with $m = 0..edgeSections - 1$, the following applies:
 - When $bS[xD_k * SubWidthC][yD_m * SubHeightC]$ is equal to 2 and $((xCb / SubWidthC + xD_k) \gg 3) \ll 3$ is equal to $xCb / SubWidthC + xD_k$, the following ordered steps apply:
 - a. The filtering process for chroma block edges as specified in clause 8.7.2.5.5 is invoked with the chroma picture sample array $recPicture_{Cb}$, the location of the chroma coding block $(xCb / SubWidthC, yCb / SubHeightC)$, the chroma location of the block (xD_k, yD_m) , a variable $edgeType$ set equal to $EDGE_VER$ and a variable $cQpPicOffset$ set equal to $pps_cb_qp_offset$ as inputs, and the modified chroma picture sample array $recPicture_{Cb}$ as output.
 - b. The filtering process for chroma block edges as specified in clause 8.7.2.5.5 is invoked with the chroma picture sample array $recPicture_{Cr}$, the location of the chroma coding block $(xCb / SubWidthC, yCb / SubHeightC)$, the chroma location of the block (xD_k, yD_m) , a variable $edgeType$ set equal to $EDGE_VER$ and a variable $cQpPicOffset$ set equal to $pps_cr_qp_offset$ as inputs, and the modified chroma picture sample array $recPicture_{Cr}$ as output.

8.7.2.5.2 Horizontal edge filtering process

Inputs to this process are:

- the picture sample array $recPicture_L$ and, when $ChromaArrayType$ is not equal to 0, the arrays $recPicture_{Cb}$ and $recPicture_{Cr}$,
- a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a variable $\log_2 CbSize$ specifying the size of the current luma coding block,
- an array bS specifying the boundary filtering strength.

Outputs of this process are the modified picture sample array $recPicture_L$ and, when $ChromaArrayType$ is not equal to 0, the arrays $recPicture_{Cb}$ and $recPicture_{Cr}$.

The filtering process for edges in the luma coding block of the current coding unit consists of the following ordered steps:

1. The variable nD is set equal to $1 \ll (\log_2 CbSize - 3)$.
2. For yD_m equal to $m \ll 3$ with $m = 0..nD - 1$ and xD_k equal to $k \ll 2$ with $k = 0..nD * 2 - 1$, the following applies:
 - When $bS[xD_k][yD_m]$ is greater than 0, the following ordered steps apply:

- a. The decision process for luma block edges as specified in clause 8.7.2.5.3 is invoked with the luma picture sample array recPicture_L , the location of the luma coding block (x_{Cb}, y_{Cb}), the luma location of the block (x_{D_k}, y_{D_m}), a variable edgeType set equal to EDGE_HOR and the boundary filtering strength $\text{bS}[x_{D_k}][y_{D_m}]$ as inputs, and the decisions dE , dEp and dEq , and the variables β and t_C as outputs.
- b. The filtering process for luma block edges as specified in clause 8.7.2.5.4 is invoked with the luma picture sample array recPicture_L , the location of the luma coding block (x_{Cb}, y_{Cb}), the luma location of the block (x_{D_k}, y_{D_m}), a variable edgeType set equal to EDGE_HOR , the decisions dEp , dEp and dEq , and the variables β and t_C as inputs, and the modified luma picture sample array recPicture_L as output.

When ChromaArrayType is not equal to 0, the following applies.

The filtering process for edges in the chroma coding blocks of current coding unit consists of the following ordered steps:

1. The variable nD is set equal to $1 \ll (\log_2 \text{CbSize} - 3)$.
2. The variable edgeSpacing is set equal to $8 / \text{SubHeightC}$.
3. The variable edgeSections is set equal to $nD * (2 / \text{SubWidthC})$.
4. For y_{D_m} equal to $m * \text{edgeSpacing}$ with $m = 0..nD - 1$ and x_{D_k} equal to $k \ll 2$ with $k = 0..\text{edgeSections} - 1$, the following applies:
 - When $\text{bS}[x_{D_k} * \text{SubWidthC}][y_{D_m} * \text{SubHeightC}]$ is equal to 2 and $((y_{Cb} / \text{SubHeightC} + y_{D_m}) \gg 3) \ll 3$ is equal to $y_{Cb} / \text{SubHeightC} + y_{D_m}$, the following ordered steps apply:
 - a. The filtering process for chroma block edges as specified in clause 8.7.2.5.5 is invoked with the chroma picture sample array recPicture_{Cb} , the location of the chroma coding block ($x_{Cb} / \text{SubWidthC}, y_{Cb} / \text{SubHeightC}$), the chroma location of the block (x_{D_k}, y_{D_m}), a variable edgeType set equal to EDGE_HOR and a variable cQpPicOffset set equal to pps_cb_qp_offset as inputs, and the modified chroma picture sample array recPicture_{Cb} as output.
 - b. The filtering process for chroma block edges as specified in clause 8.7.2.5.5 is invoked with the chroma picture sample array recPicture_{Cr} , the location of the chroma coding block ($x_{Cb} / \text{SubWidthC}, y_{Cb} / \text{SubHeightC}$), the chroma location of the block (x_{D_k}, y_{D_m}), a variable edgeType set equal to EDGE_HOR and a variable cQpPicOffset set equal to pps_cr_qp_offset as inputs, and the modified chroma picture sample array recPicture_{Cr} as output.

8.7.2.5.3 Decision process for luma block edges

Inputs to this process are:

- a luma picture sample array recPicture_L ,
- a luma location (x_{Cb}, y_{Cb}) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a luma location (x_{Bl}, y_{Bl}) specifying the top-left sample of the current luma block relative to the top-left sample of the current luma coding block,
- a variable edgeType specifying whether a vertical (EDGE_VER) or a horizontal (EDGE_HOR) edge is filtered,
- a variable bS specifying the boundary filtering strength.

Outputs of this process are:

- the variables dE , dEp and dEq containing decisions,
- the variables β and t_C .

If edgeType is equal to EDGE_VER , the sample values $p_{i,k}$ and $q_{i,k}$ with $i = 0..3$ and $k = 0$ and 3 are derived as follows:

$$q_{i,k} = \text{recPicture}_L[x_{Cb} + x_{Bl} + i][y_{Cb} + y_{Bl} + k] \quad (8-343)$$

$$p_{i,k} = \text{recPicture}_L[x_{Cb} + x_{Bl} - i - 1][y_{Cb} + y_{Bl} + k] \quad (8-344)$$

Otherwise (edgeType is equal to EDGE_HOR), the sample values $p_{i,k}$ and $q_{i,k}$ with $i = 0..3$ and $k = 0$ and 3 are derived as follows:

$$q_{i,k} = \text{recPicture}_L[\text{xCb} + \text{xBl} + k][\text{yCb} + \text{yBl} + i] \quad (8-345)$$

$$p_{i,k} = \text{recPicture}_L[\text{xCb} + \text{xBl} + k][\text{yCb} + \text{yBl} - i - 1] \quad (8-346)$$

The variables Qp_Q and Qp_P are set equal to the Qp_Y values of the coding units which include the coding blocks containing the sample $q_{0,0}$ and $p_{0,0}$, respectively.

The variable qP_L is derived as follows:

$$qP_L = ((Qp_Q + Qp_P + 1) \gg 1) \quad (8-347)$$

The value of the variable β' is determined as specified in Table 8-12 based on the luma quantization parameter Q derived as follows:

$$Q = \text{Clip3}(0, 51, qP_L + (\text{slice_beta_offset_div2} \ll 1)) \quad (8-348)$$

where $\text{slice_beta_offset_div2}$ is the value of the syntax element $\text{slice_beta_offset_div2}$ for the slice that contains sample $q_{0,0}$.

The variable β is derived as follows:

$$\beta = \beta' * (1 \ll (\text{BitDepth}_Y - 8)) \quad (8-349)$$

The value of the variable t_C' is determined as specified in Table 8-12 based on the luma quantization parameter Q derived as follows:

$$Q = \text{Clip3}(0, 53, qP_L + 2 * (bS - 1) + (\text{slice_tc_offset_div2} \ll 1)) \quad (8-350)$$

where $\text{slice_tc_offset_div2}$ is the value of the syntax element $\text{slice_tc_offset_div2}$ for the slice that contains sample $q_{0,0}$.

The variable t_C is derived as follows:

$$t_C = t_C' * (1 \ll (\text{BitDepth}_Y - 8)) \quad (8-351)$$

Depending on the value of edgeType, the following applies:

– If edgeType is equal to EDGE_VER, the following ordered steps apply:

1. The variables $dpq0$, $dpq3$, dp , dq and d are derived as follows:

$$dp0 = \text{Abs}(p_{2,0} - 2 * p_{1,0} + p_{0,0}) \quad (8-352)$$

$$dp3 = \text{Abs}(p_{2,3} - 2 * p_{1,3} + p_{0,3}) \quad (8-353)$$

$$dq0 = \text{Abs}(q_{2,0} - 2 * q_{1,0} + q_{0,0}) \quad (8-354)$$

$$dq3 = \text{Abs}(q_{2,3} - 2 * q_{1,3} + q_{0,3}) \quad (8-355)$$

$$dpq0 = dp0 + dq0 \quad (8-356)$$

$$dpq3 = dp3 + dq3 \quad (8-357)$$

$$dp = dp0 + dp3 \quad (8-358)$$

$$dq = dq0 + dq3 \quad (8-359)$$

$$d = dpq0 + dpq3 \quad (8-360)$$

2. The variables dE , dEp and dEq are set equal to 0.

3. When d is less than β , the following ordered steps apply:

- a. The variable dpq is set equal to $2 * dpq0$.
 - b. For the sample location ($x_{Cb} + x_{Bl}$, $y_{Cb} + y_{Bl}$), the decision process for a luma sample as specified in clause 8.7.2.5.6 is invoked with sample values $p_{0,0}$, $p_{3,0}$, $q_{0,0}$, and $q_{3,0}$, the variables dpq , β and t_C as inputs, and the output is assigned to the decision $dSam0$.
 - c. The variable dpq is set equal to $2 * dpq3$.
 - d. For the sample location ($x_{Cb} + x_{Bl}$, $y_{Cb} + y_{Bl} + 3$), the decision process for a luma sample as specified in clause 8.7.2.5.6 is invoked with sample values $p_{0,3}$, $p_{3,3}$, $q_{0,3}$, and $q_{3,3}$, the variables dpq , β and t_C as inputs, and the output is assigned to the decision $dSam3$.
 - e. The variable dE is set equal to 1.
 - f. When $dSam0$ is equal to 1 and $dSam3$ is equal to 1, the variable dE is set equal to 2.
 - g. When dp is less than $(\beta + (\beta \gg 1)) \gg 3$, the variable dEp is set equal to 1.
 - h. When dq is less than $(\beta + (\beta \gg 1)) \gg 3$, the variable dEq is set equal to 1.
- Otherwise ($edgeType$ is equal to $EDGE_HOR$), the following ordered steps apply:
1. The variables $dpq0$, $dpq3$, dp , dq and d are derived as follows:

$$dp0 = Abs(p_{2,0} - 2 * p_{1,0} + p_{0,0}) \quad (8-361)$$

$$dp3 = Abs(p_{2,3} - 2 * p_{1,3} + p_{0,3}) \quad (8-362)$$

$$dq0 = Abs(q_{2,0} - 2 * q_{1,0} + q_{0,0}) \quad (8-363)$$

$$dq3 = Abs(q_{2,3} - 2 * q_{1,3} + q_{0,3}) \quad (8-364)$$

$$dpq0 = dp0 + dq0 \quad (8-365)$$

$$dpq3 = dp3 + dq3 \quad (8-366)$$

$$dp = dp0 + dp3 \quad (8-367)$$

$$dq = dq0 + dq3 \quad (8-368)$$

$$d = dpq0 + dpq3 \quad (8-369)$$
 2. The variables dE , dEp and dEq are set equal to 0.
 3. When d is less than β , the following ordered steps apply:
 - a. The variable dpq is set equal to $2 * dpq0$.
 - b. For the sample location ($x_{Cb} + x_{Bl}$, $y_{Cb} + y_{Bl}$), the decision process for a luma sample as specified in clause 8.7.2.5.6 is invoked with sample values $p_{0,0}$, $p_{3,0}$, $q_{0,0}$ and $q_{3,0}$, the variables dpq , β and t_C as inputs, and the output is assigned to the decision $dSam0$.
 - c. The variable dpq is set equal to $2 * dpq3$.
 - d. For the sample location ($x_{Cb} + x_{Bl} + 3$, $y_{Cb} + y_{Bl}$), the decision process for a luma sample as specified in clause 8.7.2.5.6 is invoked with sample values $p_{0,3}$, $p_{3,3}$, $q_{0,3}$ and $q_{3,3}$, the variables dpq , β and t_C as inputs, and the output is assigned to the decision $dSam3$.
 - e. The variable dE is set equal to 1.
 - f. When $dSam0$ is equal to 1 and $dSam3$ is equal to 1, the variable dE is set equal to 2.
 - g. When dp is less than $(\beta + (\beta \gg 1)) \gg 3$, the variable dEp is set equal to 1.
 - h. When dq is less than $(\beta + (\beta \gg 1)) \gg 3$, the variable dEq is set equal to 1.

Table 8-12 – Derivation of threshold variables β' and t_c' from input Q

Q	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
β'	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	7	8
t_c'	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Q	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
β'	9	10	11	12	13	14	15	16	17	18	20	22	24	26	28	30	32	34	36
t_c'	1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4
Q	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53			
β'	38	40	42	44	46	48	50	52	54	56	58	60	62	64	-	-			
t_c'	5	5	6	6	7	8	9	10	11	13	14	16	18	20	22	24			

8.7.2.5.4 Filtering process for luma block edges

Inputs to this process are:

- a luma picture sample array $recPicture_L$,
- a luma location (x_{Cb} , y_{Cb}) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a luma location (x_{Bl} , y_{Bl}) specifying the top-left sample of the current luma block relative to the top-left sample of the current luma coding block,
- a variable $edgeType$ specifying whether a vertical (EDGE_VER) or a horizontal (EDGE_HOR) edge is filtered,
- the variables dE , dEp and dEq containing decisions,
- the variables β and t_c .

Output of this process is the modified luma picture sample array $recPicture_L$.

Depending on the value of $edgeType$, the following applies:

- If $edgeType$ is equal to EDGE_VER, the following ordered steps apply:

1. The sample values $p_{i,k}$ and $q_{i,k}$ with $i = 0..3$ and $k = 0..3$ are derived as follows:

$$q_{i,k} = recPicture_L[x_{Cb} + x_{Bl} + i][y_{Cb} + y_{Bl} + k] \quad (8-370)$$

$$p_{i,k} = recPicture_L[x_{Cb} + x_{Bl} - i - 1][y_{Cb} + y_{Bl} + k] \quad (8-371)$$

2. When dE is not equal to 0, for each sample location ($x_{Cb} + x_{Bl}$, $y_{Cb} + y_{Bl} + k$), $k = 0..3$, the following ordered steps apply:

- a. The filtering process for a luma sample as specified in clause 8.7.2.5.7 is invoked with the sample values $p_{i,k}$, $q_{i,k}$ with $i = 0..3$, the locations (x_{Pi} , y_{Pi}) set equal to ($x_{Cb} + x_{Bl} - i - 1$, $y_{Cb} + y_{Bl} + k$) and (x_{Qi} , y_{Qi}) set equal to ($x_{Cb} + x_{Bl} + i$, $y_{Cb} + y_{Bl} + k$) with $i = 0..2$, the decision dE , the variables dEp and dEq and the variable t_c as inputs, and the number of filtered samples nDp and nDq from each side of the block boundary and the filtered sample values p_i' and q_j' as outputs.

- b. When nDp is greater than 0, the filtered sample values p_i' with $i = 0..nDp - 1$ replace the corresponding samples inside the sample array $recPicture_L$ as follows:

$$recPicture_L[x_{Cb} + x_{Bl} - i - 1][y_{Cb} + y_{Bl} + k] = p_i' \quad (8-372)$$

- c. When nDq is greater than 0, the filtered sample values q_j' with $j = 0..nDq - 1$ replace the corresponding samples inside the sample array $recPicture_L$ as follows:

$$recPicture_L[x_{Cb} + x_{Bl} + j][y_{Cb} + y_{Bl} + k] = q_j' \quad (8-373)$$

- Otherwise ($edgeType$ is equal to EDGE_HOR), the following ordered steps apply:

1. The sample values $p_{i,k}$ and $q_{i,k}$ with $i = 0..3$ and $k = 0..3$ are derived as follows:

$$q_{i,k} = \text{recPicture}_L[\text{xCb} + \text{xBl} + k][\text{yCb} + \text{yBl} + i] \quad (8-374)$$

$$p_{i,k} = \text{recPicture}_L[\text{xCb} + \text{xBl} + k][\text{yCb} + \text{yBl} - i - 1] \quad (8-375)$$

2. When dE is not equal to 0, for each sample location $(\text{xCb} + \text{xBl} + k, \text{yCb} + \text{yBl})$, $k = 0..3$, the following ordered steps apply:
 - a. The filtering process for a luma sample as specified in clause 8.7.2.5.7 is invoked with the sample values $p_{i,k}$, $q_{i,k}$ with $i = 0..3$, the locations $(\text{xP}_i, \text{yP}_i)$ set equal to $(\text{xCb} + \text{xBl} + k, \text{yCb} + \text{yBl} - i - 1)$ and $(\text{xQ}_i, \text{yQ}_i)$ set equal to $(\text{xCb} + \text{xBl} + k, \text{yCb} + \text{yBl} + i)$ with $i = 0..2$, the decision dE , the variables dEp and dEq , and the variable t_c as inputs, and the number of filtered samples nDp and nDq from each side of the block boundary and the filtered sample values p'_i and q'_j as outputs.
 - b. When nDp is greater than 0, the filtered sample values p'_i with $i = 0..nDp - 1$ replace the corresponding samples inside the sample array recPicture_L as follows:

$$\text{recPicture}_L[\text{xCb} + \text{xBl} + k][\text{yCb} + \text{yBl} - i - 1] = p'_i \quad (8-376)$$

- c. When nDq is greater than 0, the filtered sample values q'_j with $j = 0..nDq - 1$ replace the corresponding samples inside the sample array recPicture_L as follows:

$$\text{recPicture}_L[\text{xCb} + \text{xBl} + k][\text{yCb} + \text{yBl} + j] = q'_j \quad (8-377)$$

8.7.2.5.5 Filtering process for chroma block edges

This process is only invoked when ChromaArrayType is not equal to 0.

Inputs to this process are:

- a chroma picture sample array s' ,
- a chroma location (xCb, yCb) specifying the top-left sample of the current chroma coding block relative to the top-left chroma sample of the current picture,
- a chroma location (xBl, yBl) specifying the top-left sample of the current chroma block relative to the top-left sample of the current chroma coding block,
- a variable edgeType specifying whether a vertical (EDGE_VER) or a horizontal (EDGE_HOR) edge is filtered,
- a variable cQpPicOffset specifying the picture-level chroma quantization parameter offset.

Output of this process is the modified chroma picture sample array s' .

If edgeType is equal to EDGE_VER , the values p_i and q_i with $i = 0..1$ and $k = 0..3$ are derived as follows:

$$q_{i,k} = s'[\text{xCb} + \text{xBl} + i][\text{yCb} + \text{yBl} + k] \quad (8-378)$$

$$p_{i,k} = s'[\text{xCb} + \text{xBl} - i - 1][\text{yCb} + \text{yBl} + k] \quad (8-379)$$

Otherwise (edgeType is equal to EDGE_HOR), the sample values p_i and q_i with $i = 0..1$ and $k = 0..3$ are derived as follows:

$$q_{i,k} = s'[\text{xCb} + \text{xBl} + k][\text{yCb} + \text{yBl} + i] \quad (8-380)$$

$$p_{i,k} = s'[\text{xCb} + \text{xBl} + k][\text{yCb} + \text{yBl} - i - 1] \quad (8-381)$$

The variables Qp_Q and Qp_P are set equal to the Qp_Y values of the coding units which include the coding blocks containing the sample $q_{0,0}$ and $p_{0,0}$, respectively.

The index qPi is derived as follows:

$$qPi = ((Qp_Q + Qp_P + 1) \gg 1) + \text{cQpPicOffset} \quad (8-382)$$

The variable Qp_C is derived as follows:

- If ChromaArrayType is equal to 1, the variable Qp_C is determined based on qPi as specified in Table 8-10.
- Otherwise (ChromaArrayType is greater than 1), the variable Qp_C is set equal to $\text{Min}(qPi, 51)$.

NOTE – The variable `cQpPicOffset` provides an adjustment for the value of `pps_cb_qp_offset` or `pps_cr_qp_offset`, according to whether the filtered chroma component is the Cb or Cr component. However, to avoid the need to vary the amount of the adjustment within the picture, the filtering process does not include an adjustment for the value of `slice_cb_qp_offset` or `slice_cr_qp_offset`, nor (when `chroma_qp_offset_list_enabled_flag` is equal to 1) for the value of `CuQpOffsetCb` or `CuQpOffsetCr`.

The value of the variable t_c' is determined as specified in Table 8-12 based on the chroma quantization parameter Q derived as follows:

$$Q = \text{Clip3}(0, 53, Q_{pC} + 2 + (\text{slice_tc_offset_div2} \ll 1)) \quad (8-383)$$

where `slice_tc_offset_div2` is the value of the syntax element `slice_tc_offset_div2` for the slice that contains sample $q_{0,0}$.

The variable t_c is derived as follows:

$$t_c = t_c' * (1 \ll (\text{BitDepth}_C - 8)) \quad (8-384)$$

Depending on the value of `edgeType`, the following applies:

- If `edgeType` is equal to `EDGE_VER`, for each sample location $(x_{Cb} + x_{Bl}, y_{Cb} + y_{Bl} + k)$, $k = 0..3$, the following ordered steps apply:

1. The filtering process for a chroma sample as specified in clause 8.7.2.5.8 is invoked with the sample values $p_{i,k}$, $q_{i,k}$, with $i = 0..1$, the locations $(x_{Cb} + x_{Bl} - 1, y_{Cb} + y_{Bl} + k)$ and $(x_{Cb} + x_{Bl}, y_{Cb} + y_{Bl} + k)$ and the variable t_c as inputs, and the filtered sample values p_0' and q_0' as outputs.
2. The filtered sample values p_0' and q_0' replace the corresponding samples inside the sample array s' as follows:

$$s'[x_{Cb} + x_{Bl}][y_{Cb} + y_{Bl} + k] = q_0' \quad (8-385)$$

$$s'[x_{Cb} + x_{Bl} - 1][y_{Cb} + y_{Bl} + k] = p_0' \quad (8-386)$$

- Otherwise (`edgeType` is equal to `EDGE_HOR`), for each sample location $(x_{Cb} + x_{Bl} + k, y_{Cb} + y_{Bl})$, $k = 0..3$, the following ordered steps apply:

1. The filtering process for a chroma sample as specified in clause 8.7.2.5.8 is invoked with the sample values $p_{i,k}$, $q_{i,k}$, with $i = 0..1$, the locations $(x_{Cb} + x_{Bl} + k, y_{Cb} + y_{Bl} - 1)$ and $(x_{Cb} + x_{Bl} + k, y_{Cb} + y_{Bl})$, and the variable t_c as inputs, and the filtered sample values p_0' and q_0' as outputs.
2. The filtered sample values p_0' and q_0' replace the corresponding samples inside the sample array s' as follows:

$$s'[x_{Cb} + x_{Bl} + k][y_{Cb} + y_{Bl}] = q_0' \quad (8-387)$$

$$s'[x_{Cb} + x_{Bl} + k][y_{Cb} + y_{Bl} - 1] = p_0' \quad (8-388)$$

8.7.2.5.6 Decision process for a luma sample

Inputs to this process are:

- the sample values p_0 , p_3 , q_0 and q_3 ,
- the variables dpq , β and t_c .

Output of this process is the variable $dSam$ containing a decision.

The variable $dSam$ is specified as follows:

- If dpq is less than $(\beta \gg 2)$, $\text{Abs}(p_3 - p_0) + \text{Abs}(q_0 - q_3)$ is less than $(\beta \gg 3)$ and $\text{Abs}(p_0 - q_0)$ is less than $(5 * t_c + 1) \gg 1$, $dSam$ is set equal to 1.
- Otherwise, $dSam$ is set equal to 0.

8.7.2.5.7 Filtering process for a luma sample

Inputs to this process are:

- the luma sample values p_i and q_i with $i = 0..3$,
- the luma locations of p_i and q_i , (x_{Pi}, y_{Pi}) and (x_{Qi}, y_{Qi}) with $i = 0..2$,
- a variable dE ,

- the variables dEp and dEq containing decisions to filter samples p1 and q1, respectively,
- a variable tC.

Outputs of this process are:

- the number of filtered samples nDp and nDq,
- the filtered sample values p_i' and q_j' with i = 0..nDp – 1, j = 0..nDq – 1.

Depending on the value of dE, the following applies:

- If the variable dE is equal to 2, nDp and nDq are both set equal to 3 and the following strong filtering applies:

$$p_0' = \text{Clip3}(p_0 - 2 * t_C, p_0 + 2 * t_C, (p_2 + 2 * p_1 + 2 * p_0 + 2 * q_0 + q_1 + 4) \gg 3) \quad (8-389)$$

$$p_1' = \text{Clip3}(p_1 - 2 * t_C, p_1 + 2 * t_C, (p_2 + p_1 + p_0 + q_0 + 2) \gg 2) \quad (8-390)$$

$$p_2' = \text{Clip3}(p_2 - 2 * t_C, p_2 + 2 * t_C, (2 * p_3 + 3 * p_2 + p_1 + p_0 + q_0 + 4) \gg 3) \quad (8-391)$$

$$q_0' = \text{Clip3}(q_0 - 2 * t_C, q_0 + 2 * t_C, (p_1 + 2 * p_0 + 2 * q_0 + 2 * q_1 + q_2 + 4) \gg 3) \quad (8-392)$$

$$q_1' = \text{Clip3}(q_1 - 2 * t_C, q_1 + 2 * t_C, (p_0 + q_0 + q_1 + q_2 + 2) \gg 2) \quad (8-393)$$

$$q_2' = \text{Clip3}(q_2 - 2 * t_C, q_2 + 2 * t_C, (p_0 + q_0 + q_1 + 3 * q_2 + 2 * q_3 + 4) \gg 3) \quad (8-394)$$

- Otherwise, nDp and nDq are set both equal to 0 and the following weak filtering applies:

- The following applies:

$$\Delta = (9 * (q_0 - p_0) - 3 * (q_1 - p_1) + 8) \gg 4 \quad (8-395)$$

- When Abs(Δ) is less than tC * 10, the following ordered steps apply:

1. The filtered sample values p₀' and q₀' are specified as follows:

$$\Delta = \text{Clip3}(-t_C, t_C, \Delta) \quad (8-396)$$

$$p_0' = \text{Clip1}_Y(p_0 + \Delta) \quad (8-397)$$

$$q_0' = \text{Clip1}_Y(q_0 - \Delta) \quad (8-398)$$

2. When dEp is equal to 1, the filtered sample value p₁' is specified as follows:

$$\Delta p = \text{Clip3}(-(t_C \gg 1), t_C \gg 1, (((p_2 + p_0 + 1) \gg 1) - p_1 + \Delta) \gg 1) \quad (8-399)$$

$$p_1' = \text{Clip1}_Y(p_1 + \Delta p) \quad (8-400)$$

3. When dEq is equal to 1, the filtered sample value q₁' is specified as follows:

$$\Delta q = \text{Clip3}(-(t_C \gg 1), t_C \gg 1, (((q_2 + q_0 + 1) \gg 1) - q_1 - \Delta) \gg 1) \quad (8-401)$$

$$q_1' = \text{Clip1}_Y(q_1 + \Delta q) \quad (8-402)$$

4. nDp is set equal to dEp + 1 and nDq is set equal to dEq + 1.

When nDp is greater than 0 and one or more of the following conditions are true, nDp is set equal to 0:

- pcm_loop_filter_disabled_flag is equal to 1 and pcm_flag[xP₀][yP₀] is equal to 1.
- cu_transquant_bypass_flag of the coding unit that includes the coding block containing the sample p₀ is equal to 1.
- palette_mode_flag of the coding unit that includes the coding block containing the sample p₀ is equal to 1.

When nDq is greater than 0 and one or more of the following conditions are true, nDq is set equal to 0:

- pcm_loop_filter_disabled_flag is equal to 1 and pcm_flag[xQ₀][yQ₀] is equal to 1.
- cu_transquant_bypass_flag of the coding unit that includes the coding block containing the sample q₀ is equal to 1.
- palette_mode_flag of the coding unit that includes the coding block containing the sample q₀ is equal to 1.

8.7.2.5.8 Filtering process for a chroma sample

This process is only invoked when ChromaArrayType is not equal to 0.

Inputs to this process are:

- the chroma sample values p_i and q_i with i = 0..1,
- the chroma locations of p₀ and q₀, (xP₀, yP₀) and (xQ₀, yQ₀),
- a variable t_C.

Outputs of this process are the filtered sample values p₀' and q₀'.

The filtered sample values p₀' and q₀' are derived as follows:

$$\Delta = \text{Clip3}(-t_C, t_C, ((((q_0 - p_0) \ll 2) + p_1 - q_1 + 4) \gg 3)) \quad (8-403)$$

$$p_0' = \text{Clip1}_C(p_0 + \Delta) \quad (8-404)$$

$$q_0' = \text{Clip1}_C(q_0 - \Delta) \quad (8-405)$$

When one or more of the following conditions are true, the filtered sample value, p₀' is substituted by the corresponding input sample value p₀:

- pcm_loop_filter_disabled_flag is equal to 1 and pcm_flag[xP₀ * SubWidthC][yP₀ * SubHeightC] is equal to 1.
- cu_transquant_bypass_flag of the coding unit that includes the coding block containing the sample p₀ is equal to 1.
- palette_mode_flag of the coding unit that includes the coding block containing the sample p₀ is equal to 1.

When one or more of the following conditions are true, the filtered sample value, q₀' is substituted by the corresponding input sample value q₀:

- pcm_loop_filter_disabled_flag is equal to 1 and pcm_flag[xQ₀ * SubWidthC][yQ₀ * SubHeightC] is equal to 1.
- cu_transquant_bypass_flag of the coding unit that includes the coding block containing the sample q₀ is equal to 1.
- palette_mode_flag of the coding unit that includes the coding block containing the sample q₀ is equal to 1.

8.7.3 Sample adaptive offset process

8.7.3.1 General

Inputs to this process are the reconstructed picture sample array prior to sample adaptive offset recPicture_L and, when ChromaArrayType is not equal to 0, the arrays recPicture_{Cb} and recPicture_{Cr}.

Outputs of this process are the modified reconstructed picture sample array after sample adaptive offset saoPicture_L and, when ChromaArrayType is not equal to 0, the arrays saoPicture_{Cb} and saoPicture_{Cr}.

This process is performed on a CTB basis after the completion of the deblocking filter process for the decoded picture.

The sample values in the modified reconstructed picture sample array saoPicture_L and, when ChromaArrayType is not equal to 0, the arrays saoPicture_{Cb} and saoPicture_{Cr} are initially set equal to the sample values in the reconstructed picture sample array recPicture_L and, when ChromaArrayType is not equal to 0, the arrays recPicture_{Cb} and recPicture_{Cr}, respectively.

For every CTU with CTB location (rx, ry), where rx = 0..PicWidthInCtbsY – 1 and ry = 0..PicHeightInCtbsY – 1, the following applies:

- When slice_sao_luma_flag of the current slice is equal to 1, the CTB modification process as specified in clause 8.7.3.2 is invoked with recPicture set equal to recPicture_L, cIdx set equal to 0, (rx, ry), and both nCtbSw and nCtbSh set equal to CtbSizeY as inputs, and the modified luma picture sample array saoPicture_L as output.
- When ChromaArrayType is not equal to 0 and slice_sao_chroma_flag of the current slice is equal to 1, the CTB modification process as specified in clause 8.7.3.2 is invoked with recPicture set equal to recPicture_{Cb}, cIdx set equal

to 1, (rx, ry), nCtbSw set equal to $(1 \ll \text{CtbLog2SizeY}) / \text{SubWidthC}$ and nCtbSh set equal to $(1 \ll \text{CtbLog2SizeY}) / \text{SubHeightC}$ as inputs, and the modified chroma picture sample array saoPicture_{Cb} as output.

- When ChromaArrayType is not equal to 0 and slice_sao_chroma_flag of the current slice is equal to 1, the CTB modification process as specified in clause 8.7.3.2 is invoked with recPicture set equal to recPicture_{Cr}, cIdx set equal to 2, (rx, ry), nCtbSw set equal to $(1 \ll \text{CtbLog2SizeY}) / \text{SubWidthC}$ and nCtbSh set equal to $(1 \ll \text{CtbLog2SizeY}) / \text{SubHeightC}$ as inputs, and the modified chroma picture sample array saoPicture_{Cr} as output.

8.7.3.2 CTB modification process

Inputs to this process are:

- the picture sample array recPicture for the colour component cIdx,
- a variable cIdx specifying the colour component index,
- a pair of variables (rx, ry) specifying the CTB location,
- the CTB width nCtbSw and height nCtbSh.

Output of this process is a modified picture sample array saoPicture for the colour component cIdx.

The variable bitDepth is derived as follows:

- If cIdx is equal to 0, bitDepth is set equal to BitDepth_Y.
- Otherwise, bitDepth is set equal to BitDepth_C.

The location (xCtb, yCtb), specifying the top-left sample of the current CTB for the colour component cIdx relative to the top-left sample of the current picture component cIdx, is derived as follows:

$$(xCtb, yCtb) = (rx * nCtbSw, ry * nCtbSh) \quad (8-406)$$

The sample locations inside the current CTB are derived as follows:

$$(xS_i, yS_j) = (xCtb + i, yCtb + j) \quad (8-407)$$

$$(xY_i, yY_j) = (cIdx == 0) ? (xS_i, yS_j) : (xS_i * \text{SubWidthC}, yS_j * \text{SubHeightC}) \quad (8-408)$$

For all sample locations (xS_i, yS_j) and (xY_i, yY_j) with $i = 0..nCtbSw - 1$ and $j = 0..nCtbSh - 1$, depending on the values of pcm_loop_filter_disabled_flag, pcm_flag[xY_i][yY_j] and cu_transquant_bypass_flag of the coding unit which includes the coding block covering recPicture[xS_i][yS_j], the following applies:

- If one or more of the following conditions are true, saoPicture[xS_i][yS_j] is not modified:
 - pcm_loop_filter_disabled_flag and pcm_flag[xY_i][yY_j] are both equal to 1.
 - cu_transquant_bypass_flag is equal to 1.
 - SaoTypeIdx[cIdx][rx][ry] is equal to 0.
- Otherwise, if SaoTypeIdx[cIdx][rx][ry] is equal to 2, the following ordered steps apply:
 1. The values of hPos[k] and vPos[k] for $k = 0..1$ are specified in Table 8-13 based on SaoEoClass[cIdx][rx][ry].
 2. The variable edgeIdx is derived as follows:
 - The modified sample locations (xS_{ik}', yS_{jk}') and (xY_{ik}', yY_{jk}') are derived as follows:

$$(xS_{ik}', yS_{jk}') = (xS_i + hPos[k], yS_j + vPos[k]) \quad (8-409)$$

$$(xY_{ik}', yY_{jk}') = (cIdx == 0) ? (xS_{ik}', yS_{jk}') : (xS_{ik}' * \text{SubWidthC}, yS_{jk}' * \text{SubHeightC}) \quad (8-410)$$

- If one or more of the following conditions for all sample locations (xS_{ik}', yS_{jk}') and (xY_{ik}', yY_{jk}') with $k = 0..1$ are true, edgeIdx is set equal to 0:
 - The sample at location (xS_{ik}', yS_{jk}') is outside the picture boundaries.
 - The sample at location (xS_{ik}', yS_{jk}') belongs to a different slice and one of the following two conditions is true:

- $\text{MinTbAddrZs}[xY_{ik}' \gg \text{MinTbLog2SizeY}][yY_{jk}' \gg \text{MinTbLog2SizeY}]$ is less than $\text{MinTbAddrZs}[xY_i \gg \text{MinTbLog2SizeY}][yY_j \gg \text{MinTbLog2SizeY}]$ and $\text{slice_loop_filter_across_slices_enabled_flag}$ in the slice which the sample $\text{recPicture}[xS_i][yS_j]$ belongs to is equal to 0.
- $\text{MinTbAddrZs}[xY_i \gg \text{MinTbLog2SizeY}][yY_j \gg \text{MinTbLog2SizeY}]$ is less than $\text{MinTbAddrZs}[xY_{ik}' \gg \text{MinTbLog2SizeY}][yY_{jk}' \gg \text{MinTbLog2SizeY}]$ and $\text{slice_loop_filter_across_slices_enabled_flag}$ in the slice which the sample $\text{recPicture}[xS_{ik}'][yS_{jk}']$ belongs to is equal to 0.
- $\text{loop_filter_across_tiles_enabled_flag}$ is equal to 0 and the sample at location (xS_{ik}', yS_{jk}') belongs to a different tile.
- Otherwise, edgeIdx is derived as follows:
- The following applies:

$$\begin{aligned} \text{edgeIdx} = & \\ & 2 + \text{Sign}(\text{recPicture}[xS_i][yS_j] - \text{recPicture}[xS_i + \text{hPos}[0]][yS_j + \text{vPos}[0]]) + \\ & \text{Sign}(\text{recPicture}[xS_i][yS_j] - \text{recPicture}[xS_i + \text{hPos}[1]][yS_j + \text{vPos}[1]]) \end{aligned} \quad (8-411)$$

- When edgeIdx is equal to 0, 1, or 2, edgeIdx is modified as follows:

$$\text{edgeIdx} = (\text{edgeIdx} == 2) ? 0 : (\text{edgeIdx} + 1) \quad (8-412)$$

3. The modified picture sample array $\text{saoPicture}[xS_i][yS_j]$ is derived as follows:

$$\begin{aligned} \text{saoPicture}[xS_i][yS_j] = & \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, \text{recPicture}[xS_i][yS_j] + \\ & \text{SaoOffsetVal}[\text{cIdx}][\text{rx}][\text{ry}][\text{edgeIdx}]) \end{aligned} \quad (8-413)$$

- Otherwise ($\text{SaoTypeIdx}[\text{cIdx}][\text{rx}][\text{ry}]$ is equal to 1), the following ordered steps apply:

 1. The variable bandShift is set equal to $\text{bitDepth} - 5$.
 2. The variable saoLeftClass is set equal to $\text{sao_band_position}[\text{cIdx}][\text{rx}][\text{ry}]$.
 3. The list bandTable is defined with 32 elements and all elements are initially set equal to 0. Then, four of its elements (indicating the starting position of bands for explicit offsets) are modified as follows:

$$\begin{aligned} \text{for}(\text{k} = 0; \text{k} < 4; \text{k}++) \\ \text{bandTable}[(\text{k} + \text{saoLeftClass}) \& 31] = \text{k} + 1 \end{aligned} \quad (8-414)$$

4. The variable bandIdx is set equal to $\text{bandTable}[\text{recPicture}[xS_i][yS_j] \gg \text{bandShift}]$.
5. The modified picture sample array $\text{saoPicture}[xS_i][yS_j]$ is derived as follows:

$$\begin{aligned} \text{saoPicture}[xS_i][yS_j] = & \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, \text{recPicture}[xS_i][yS_j] + \\ & \text{SaoOffsetVal}[\text{cIdx}][\text{rx}][\text{ry}][\text{bandIdx}]) \end{aligned} \quad (8-415)$$

Table 8-13 – Specification of hPos and vPos according to the sample adaptive offset class

$\text{SaoEoClass}[\text{cIdx}][\text{rx}][\text{ry}]$	0	1	2	3
$\text{hPos}[0]$	−1	0	−1	1
$\text{hPos}[1]$	1	0	1	−1
$\text{vPos}[0]$	0	−1	−1	−1
$\text{vPos}[1]$	0	1	1	1

9 Parsing process

9.1 General

Inputs to this process are bits from the RBSP.

Outputs of this process are syntax element values.

This process is invoked when the descriptor of a syntax element in the syntax tables is equal to ue(v), se(v) (see clause 9.2), or ae(v) (see clause 9.3).

9.2 Parsing process for 0-th order Exp-Golomb codes

9.2.1 General

This process is invoked when the descriptor of a syntax element in the syntax tables is equal to ue(v) or se(v).

Inputs to this process are bits from the RBSP.

Outputs of this process are syntax element values.

Syntax elements coded as ue(v) or se(v) are Exp-Golomb-coded. The parsing process for these syntax elements begins with reading the bits starting at the current location in the bitstream up to and including the first non-zero bit, and counting the number of leading bits that are equal to 0. This process is specified as follows:

```

leadingZeroBits = -1
for( b = 0; !b; leadingZeroBits++ )
    b = read_bits( 1 )

```

(9-1)

The variable codeNum is then assigned as follows:

$$\text{codeNum} = 2^{\text{leadingZeroBits}} - 1 + \text{read_bits}(\text{leadingZeroBits}) \quad (9-2)$$

where the value returned from read_bits(leadingZeroBits) is interpreted as a binary representation of an unsigned integer with most significant bit written first.

Table 9-1 illustrates the structure of the Exp-Golomb code by separating the bit string into "prefix" and "suffix" bits. The "prefix" bits are those bits that are parsed as specified above for the computation of leadingZeroBits, and are shown as either 0 or 1 in the bit string column of Table 9-1. The "suffix" bits are those bits that are parsed in the computation of codeNum and are shown as x_i in Table 9-1, with i in the range of 0 to leadingZeroBits - 1, inclusive. Each x_i is equal to either 0 or 1.

Table 9-1 – Bit strings with "prefix" and "suffix" bits and assignment to codeNum ranges (informative)

Bit string form	Range of codeNum
1	0
0 1 x_0	1..2
0 0 1 $x_1 x_0$	3..6
0 0 0 1 $x_2 x_1 x_0$	7..14
0 0 0 0 1 $x_3 x_2 x_1 x_0$	15..30
0 0 0 0 0 1 $x_4 x_3 x_2 x_1 x_0$	31..62
...	...

Table 9-2 illustrates explicitly the assignment of bit strings to codeNum values.

Table 9-2 – Exp-Golomb bit strings and codeNum in explicit form and used as ue(v) (informative)

Bit string	codeNum
1	0
0 1 0	1
0 1 1	2
0 0 1 0 0	3
0 0 1 0 1	4
0 0 1 1 0	5
0 0 1 1 1	6
0 0 0 1 0 0 0	7
0 0 0 1 0 0 1	8
0 0 0 1 0 1 0	9
...	...

Depending on the descriptor, the value of a syntax element is derived as follows:

- If the syntax element is coded as ue(v), the value of the syntax element is equal to codeNum.
- Otherwise (the syntax element is coded as se(v)), the value of the syntax element is derived by invoking the mapping process for signed Exp-Golomb codes as specified in clause 9.2.2 with codeNum as input.

9.2.2 Mapping process for signed Exp-Golomb codes

Input to this process is codeNum as specified in clause 9.2.

Output of this process is a value of a syntax element coded as se(v).

The syntax element is assigned to the codeNum by ordering the syntax element by its absolute value in increasing order and representing the positive value for a given absolute value with the lower codeNum. Table 9-3 provides the assignment rule.

Table 9-3 – Assignment of syntax element to codeNum for signed Exp-Golomb coded syntax elements se(v)

codeNum	syntax element value
0	0
1	1
2	–1
3	2
4	–2
5	3
6	–3
k	$(-1)^{k+1} \text{Ceil}(k \div 2)$

9.3 CABAC parsing process for slice segment data

9.3.1 General

This process is invoked when parsing syntax elements with descriptor `ae(v)` in clauses 7.3.8.1 through 7.3.8.12.

Inputs to this process are a request for a value of a syntax element and values of prior parsed syntax elements.

Output of this process is the value of the syntax element.

The initialization process as specified in clause 9.3.2 is invoked when starting the parsing of one or more of the following:

1. The slice segment data syntax specified in clause 7.3.8.1,
2. The CTU syntax specified in clause 7.3.8.2 and the CTU is the first CTU in a tile,
3. The CTU syntax specified in clause 7.3.8.2, `entropy_coding_sync_enabled_flag` is equal to 1 and the associated luma CTB is the first luma CTB in a CTU row of a tile.

The parsing of syntax elements proceeds as follows:

When `cabac_bypass_alignment_enabled_flag` is equal to 1, the request for a value of a syntax element is for either the syntax elements `coeff_abs_level_remaining[]` or `coeff_sign_flag[]` and `escapeDataPresent` is equal to 1, the alignment process prior to aligned bypass decoding as specified in clause 9.3.4.3.6 is invoked.

For each requested value of a syntax element a binarization is derived as specified in clause 9.3.3.

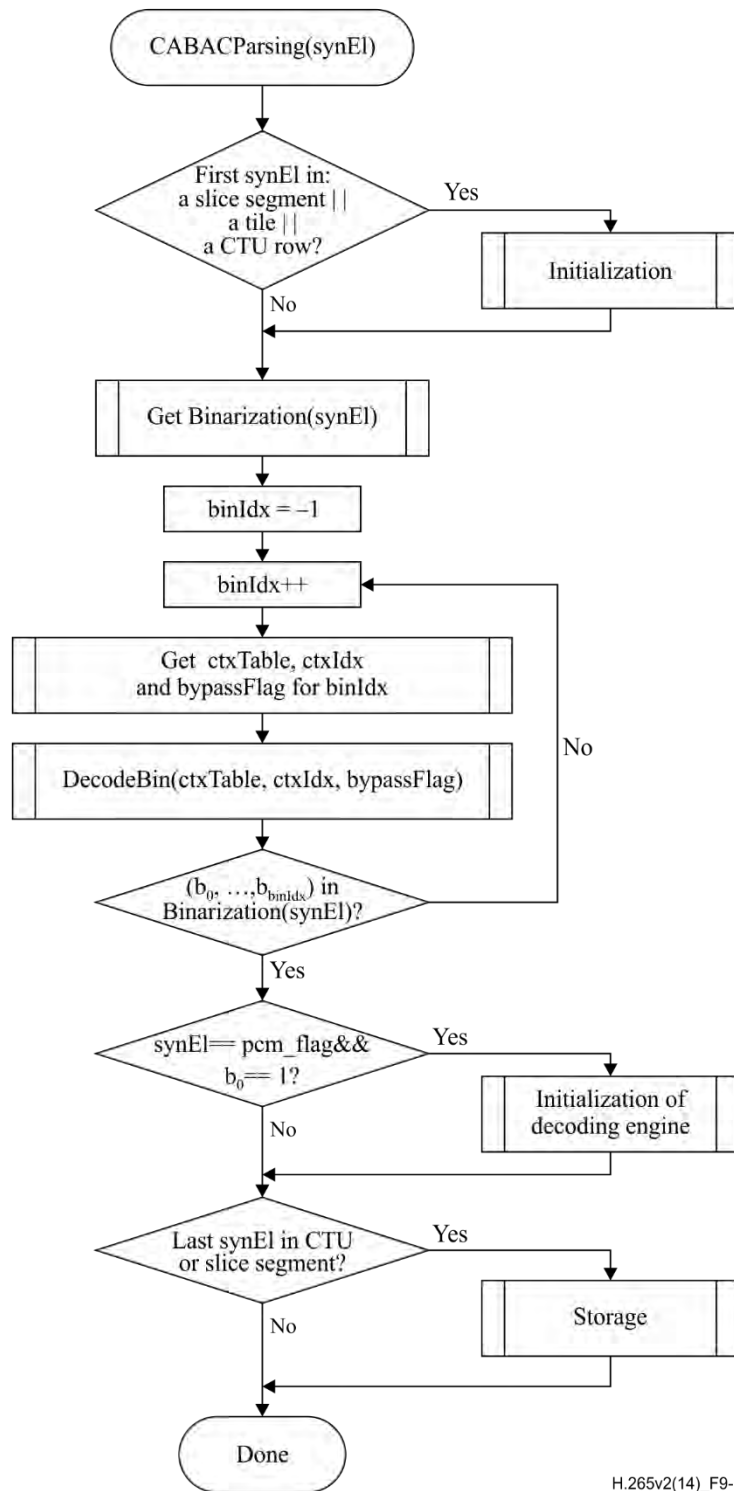
The binarization for the syntax element and the sequence of parsed bins determines the decoding process flow as described in clause 9.3.4.

In case the request for a value of a syntax element is processed for the syntax element `pcm_flag` and the decoded value of `pcm_flag` is equal to 1, the decoding engine is initialized after the decoding of any `pcm_alignment_zero_bit` and all `pcm_sample_luma` and `pcm_sample_chroma` data as specified in clause 9.3.2.6.

The storage process for context variables is applied as follows:

- When ending the parsing of the CTU syntax in clause 7.3.8.2, `entropy_coding_sync_enabled_flag` is equal to 1 and either `CtbAddrInRs % PicWidthInCtbsY` is equal to 1 or both `CtbAddrInRs` is greater than 1 and `TileId[CtbAddrInTs]` is not equal to `TileId[CtbAddrRsToTs[CtbAddrInRs – 2]]`, the storage process for context variables, Rice parameter initialization states, and palette predictor variables as specified in clause 9.3.2.4 is invoked with `TableStateIdxWpp`, `TableMpsValWpp`, `TableStatCoeffWpp` when `persistent_rice_adaptation_enabled_flag` is equal to 1, and `PredictorPaletteSizeWpp` and `PredictorPaletteEntriesWpp` when `palette_mode_enabled_flag` is equal to 1 as outputs.
- When ending the parsing of the general slice segment data syntax in clause 7.3.8.1, `dependent_slice_segments_enabled_flag` is equal to 1 and `end_of_slice_segment_flag` is equal to 1, the storage process for context variables, Rice parameter initialization states, and palette predictor variables as specified in clause 9.3.2.4 is invoked with `TableStateIdxDs`, `TableMpsValDs`, `TableStatCoeffDs` when `persistent_rice_adaptation_enabled_flag` is equal to 1, and `PredictorPaletteSizeDs` and `PredictorPaletteEntriesDs` when `palette_mode_enabled_flag` is equal to 1 as outputs.

The whole CABAC parsing process for a syntax element `synEl` is illustrated in Figure 9-1.



H.265v2(14)_F9-1

Figure 9-1 – Illustration of CABAC parsing process for a syntax element synEl (informative)

9.3.2 Initialization process

9.3.2.1 General

Outputs of this process are initialized CABAC internal variables, the initialized Rice parameter initialization states StatCoeff, and the initialized palette predictor variables.

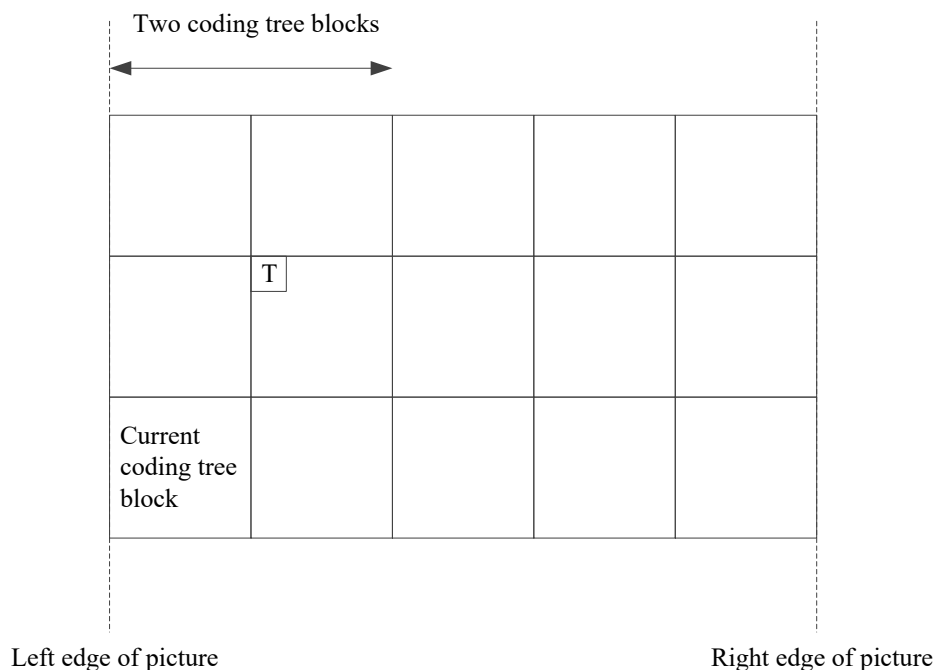


Figure 9-2 – Spatial neighbour T that is used to invoke the CTB availability derivation process relative to the current CTB (informative)

The context variables of the arithmetic decoding engine, Rice parameter initialization states, and palette predictor variables are initialized as follows:

- If the CTU is the first CTU in a tile, the following applies:
 - The initialization process for context variables is invoked as specified in clause 9.3.2.2.
 - The variables StatCoeff[k] are set equal to 0, for k in the range 0 to 3, inclusive.
 - The initialization process for palette predictor variables is invoked as specified in clause 9.3.2.3.
- Otherwise, if entropy_coding_sync_enabled_flag is equal to 1 and either CtbAddrInRs % PicWidthInCtbsY is equal to 0 or TileId[CtbAddrInTs] is not equal to TileId[CtbAddrRsToTs[CtbAddrInRs – 1]], the following applies:
 - The location (xNbT, yNbT) of the top-left luma sample of the spatial neighbouring block T (Figure 9-2) is derived using the location (x0, y0) of the top-left luma sample of the current CTB as follows:

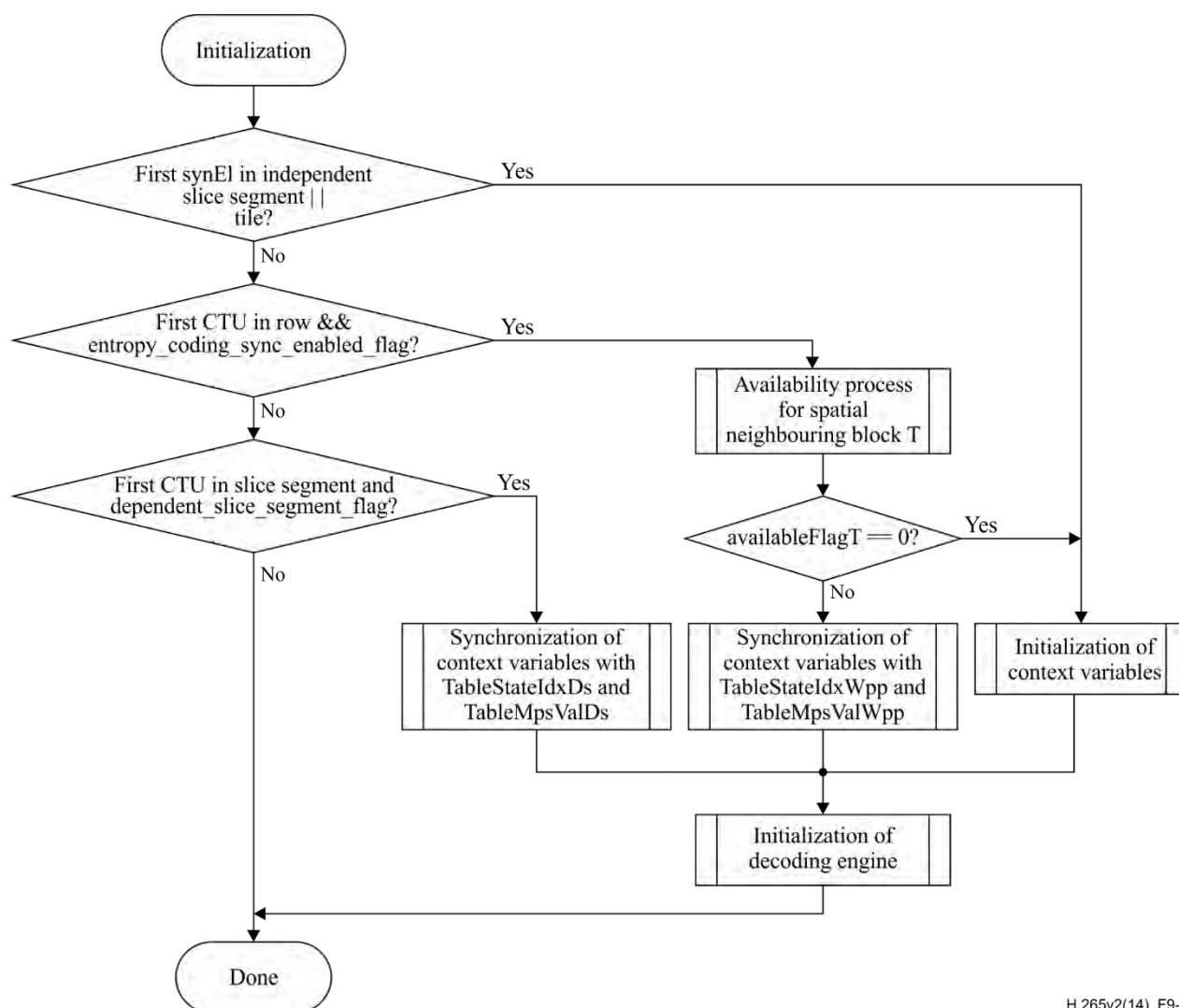
$$(xNbT, yNbT) = (x0 + CtbSizeY, y0 - CtbSizeY) \quad (9-3)$$

- The availability derivation process for a block in z-scan order as specified in clause 6.4.1 is invoked with the location (xCurr, yCurr) set equal to (x0, y0) and the neighbouring location (xNbY, yNbY) set equal to (xNbT, yNbT) as inputs, and the output is assigned to availableFlagT.
- The synchronization process for context variables, Rice parameter initialization states, and palette predictor variables is invoked as follows:
 - If availableFlagT is equal to 1, the synchronization process for context variables, Rice parameter initialization states, and palette predictor variables as specified in clause 9.3.2.5 is invoked with TableStateIdxWpp, TableMpsValWpp, TableStatCoeffWpp, PredictorPaletteSizeWpp, and TablePredictorPaletteEntriesWpp as inputs.
 - Otherwise, the following applies:
 - The initialization process for context variables is invoked as specified in clause 9.3.2.2.

- The variables StatCoeff[k] are set equal to 0, for k in the range 0 to 3, inclusive.
- The initialization process for palette predictor variables is invoked as specified in clause 9.3.2.3.
- Otherwise, if CtbAddrInRs is equal to slice_segment_address and dependent_slice_segment_flag is equal to 1, the synchronization process for context variables and Rice parameter initialization states as specified in clause 9.3.2.5 is invoked with TableStateIdxDs, TableMpsValDs, TableStatCoeffDs, PredictorPaletteSizeDs, and TablePredictorPaletteEntriesDs as inputs.
- Otherwise, the following applies:
 - The initialization process for context variables is invoked as specified in clause 9.3.2.2.
 - The variables StatCoeff[k] are set equal to 0, for k in the range 0 to 3, inclusive.
 - The initialization process for palette predictor variables is invoked as specified in clause 9.3.2.3.

The initialization process for the arithmetic decoding engine is invoked as specified in clause 9.3.2.6.

The whole initialization process for a syntax element synEl is illustrated in the flowchart of Figure 9-3.



H.265v2(14)_F9-3

Figure 9-3 – Illustration of CABAC initialization process (informative)

9.3.2.2 Initialization process for context variables

Outputs of this process are the initialized CABAC context variables indexed by ctxTable and ctxIdx.

Table 9-5 to Table 9-37 contain the values of the 8 bit variable initValue used in the initialization of context variables that are assigned to all syntax elements in clauses 7.3.8.1 through 7.3.8.12, except end_of_slice_segment_flag, end_of_subset_one_bit and pcm_flag.

For each context variable, the two variables pStateIdx and valMps are initialized.

NOTE 1 – The variable pStateIdx corresponds to a probability state index and the variable valMps corresponds to the value of the most probable symbol as further described in clause 9.3.4.3.

From the 8 bit table entry initValue, the two 4 bit variables slopeIdx and offsetIdx are derived as follows:

$$\begin{aligned} \text{slopeIdx} &= \text{initValue} \gg 4 \\ \text{offsetIdx} &= \text{initValue} \& 15 \end{aligned} \quad (9-4)$$

The variables m and n, used in the initialization of context variables, are derived from slopeIdx and offsetIdx as follows:

$$\begin{aligned} m &= \text{slopeIdx} * 5 - 45 \\ n &= (\text{offsetIdx} \ll 3) - 16 \end{aligned} \quad (9-5)$$

The two values assigned to pStateIdx and valMps for the initialization are derived from SliceQp_V, which is derived in Equation 7-54. Given the variables m and n, the initialization is specified as follows:

$$\begin{aligned} \text{preCtxState} &= \text{Clip3}(1, 126, ((m * \text{Clip3}(0, 51, \text{SliceQp}_V)) \gg 4) + n) \\ \text{valMps} &= (\text{preCtxState} \leq 63) ? 0 : 1 \\ \text{pStateIdx} &= \text{valMps} ? (\text{preCtxState} - 64) : (63 - \text{preCtxState}) \end{aligned} \quad (9-6)$$

In Table 9-4, the ctxIdx for which initialization is needed for each of the three initialization types, specified by the variable initType, are listed. Also listed is the table number that includes the values of initValue needed for the initialization. For P and B slice types, the derivation of initType depends on the value of the cabac_init_flag syntax element. The variable initType is derived as follows:

$$\begin{aligned} &\text{if}(\text{slice_type} == \text{I}) \\ &\quad \text{initType} = 0 \\ &\text{else if}(\text{slice_type} == \text{P}) \\ &\quad \text{initType} = \text{cabac_init_flag} ? 2 : 1 \\ &\text{else} \\ &\quad \text{initType} = \text{cabac_init_flag} ? 1 : 2 \end{aligned} \quad (9-7)$$

Table 9-4 – Association of ctxIdx and syntax elements for each initializationType in the initialization process

Syntax structure	Syntax element	ctxTable	initType		
			0	1	2
sao()	sao_merge_left_flag sao_merge_up_flag	Table 9-5	0	1	2
	sao_type_idx_luma sao_type_idx_chroma	Table 9-6	0	1	2
coding_quadtree()	split_cu_flag[][]	Table 9-7	0..2	3..5	6..8
coding_unit()	cu_transquant_bypass_flag	Table 9-8	0	1	2
	cu_skip_flag	Table 9-9		0..2	3..5
	palette_mode_flag	Table 9-38	0	1	2
	pred_mode_flag	Table 9-10		0	1
	part_mode	Table 9-11	0	1..4	5..8
	prev_intra_luma_pred_flag[][]	Table 9-12	0	1	2
	intra_chroma_pred_mode[][]	Table 9-13	0	1	2
	rqt_root_cbf	Table 9-14		0	1
prediction_unit()	merge_flag[][]	Table 9-15		0	1

Table 9-4 – Association of ctxIdx and syntax elements for each initializationType in the initialization process

Syntax structure	Syntax element	ctxTable	initType		
			0	1	2
	merge_idx[][]	Table 9-16		0	1
	inter_pred_idc[][]	Table 9-17		0..4	5..9
	ref_idx_l0[][], ref_idx_l1[][]	Table 9-18		0..1	2..3
	mvp_l0_flag[][], mvp_l1_flag[][]	Table 9-19		0	1
transform_tree()	split_transform_flag[][][]	Table 9-20	0..2	3..5	6..8
	cbf_luma[][][]	Table 9-21	0..1	2..3	4..5
	cbf_cb[][][], cbf_cr[][][]	Table 9-22	0..3 12	4..7 13	8..11 14
mvd_coding()	abs_mvd_greater0_flag[]	Table 9-23		0	2
	abs_mvd_greater1_flag[]	Table 9-23		1	3
transform_unit()	tu_residual_act_flag	Table 9-39	0	1	2
cross_comp_pred()	log2_res_scale_abs_plus1[]	Table 9-36	0..7	8..15	16..23
	res_scale_sign_flag[]	Table 9-37	0..1	2..3	4..5
residual_coding()	transform_skip_flag[][][0]	Table 9-25	0	1	2
	transform_skip_flag[][][1]	Table 9-25	3	4	5
	transform_skip_flag[][][2]				
	explicit_rdpem_flag[][][0]	Table 9-32		0	1
	explicit_rdpem_flag[][][1]	Table 9-32		2	3
	explicit_rdpem_flag[][][2]				
	explicit_rdpem_dir_flag[][][0]	Table 9-33		0	1
	explicit_rdpem_dir_flag[][][1]	Table 9-33		2	3
	explicit_rdpem_dir_flag[][][2]				
	last_sig_coeff_x_prefix	Table 9-26	0..17	18..35	36..53
	last_sig_coeff_y_prefix	Table 9-27	0..17	18..35	36..53
	coded_sub_block_flag[][]	Table 9-28	0..3	4..7	8..11
	sig_coeff_flag[][]	Table 9-29	0..41 126..127	42..83 128..129	84..125 130..131
	coeff_abs_level_greater1_flag[]	Table 9-30	0..23	24..47	48..71
	coeff_abs_level_greater2_flag[]	Table 9-31	0..5	6..11	12..17
palette_coding()	palette_run_prefix	Table 9-40	0..7	8..15	16..23
	copy_above_palette_indices_flag	Table 9-41	0	1	2
	copy_above_indices_for_final_run_flag	Table 9-41	0	1	2
	palette_transpose_flag	Table 9-42	0	1	2
delta_qp()	cu_qp_delta_abs	Table 9-24	0..1	2..3	4..5
chroma_qp_offset()	cu_chroma_qp_offset_flag	Table 9-34	0	1	2
	cu_chroma_qp_offset_idx	Table 9-35	0	1	2

NOTE 2 – ctxTable equal to 0 and ctxIdx equal to 0 are associated with end_of_slice_segment_flag, end_of_subset_one_bit and pcm_flag. The decoding process specified in clause 9.3.4.3.5 applies to ctxTable equal to 0 and ctxIdx equal to 0. This decoding process, however, may also be implemented by using the decoding process specified in clause 9.3.4.3.2. In this case, the initial values associated with ctxTable equal to 0 and ctxIdx equal to 0 are specified to be pStateIdx = 63 and valMps = 0, where pStateIdx = 63 represents a non-adapting probability state.

Table 9-5 – Values of initValue for ctxIdx of sao_merge_left_flag and sao_merge_up_flag

Initialization variable	ctxIdx of sao_merge_left_flag and sao_merge_up_flag		
	0	1	2
initValue	153	153	153

Table 9-6 – Values of initValue for ctxIdx of sao_type_idx_luma and sao_type_idx_chroma

Initialization variable	ctxIdx of sao_type_idx_luma and sao_type_idx_chroma		
	0	1	2
initValue	200	185	160

Table 9-7 – Values of initValue for ctxIdx of split_cu_flag

Initialization variable	ctxIdx of split_cu_flag								
	0	1	2	3	4	5	6	7	8
initValue	139	141	157	107	139	126	107	139	126

Table 9-8 – Values of initValue for ctxIdx of cu_transquant_bypass_flag

Initialization variable	ctxIdx of cu_transquant_bypass_flag		
	0	1	2
initValue	154	154	154

Table 9-9 – Values of initValue for ctxIdx of cu_skip_flag

Initialization variable	ctxIdx of cu_skip_flag					
	0	1	2	3	4	5
initValue	197	185	201	197	185	201

Table 9-10 – Values of initValue for ctxIdx of pred_mode_flag

Initialization variable	ctxIdx of pred_mode_flag	
	0	1
initValue	149	134

Table 9-11 – Values of initValue for ctxIdx of part_mode

Initialization variable	ctxIdx of part_mode								
	0	1	2	3	4	5	6	7	8
initValue	184	154	139	154	154	154	139	154	154

Table 9-12 – Values of initValue for ctxIdx of prev_intra_luma_pred_flag

Initialization variable	ctxIdx of prev_intra_luma_pred_flag		
	0	1	2
initValue	184	154	183

Table 9-13 – Values of initValue for ctxIdx of intra_chroma_pred_mode

Initialization variable	ctxIdx of intra_chroma_pred_mode		
	0	1	2
initValue	63	152	152

Table 9-14 – Values of initValue for ctxIdx of rqt_root_cbf

Initialization variable	ctxIdx of rqt_root_cbf	
	0	1
initValue	79	79

Table 9-15 – Values of initValue for ctxIdx of merge_flag

Initialization variable	ctxIdx of merge_flag	
	0	1
initValue	110	154

Table 9-16 – Values of initValue for ctxIdx of merge_idx

Initialization variable	ctxIdx of merge_idx	
	0	1
initValue	122	137

Table 9-17 – Values of initValue for ctxIdx of inter_pred_idc

Initialization variable	ctxIdx of inter_pred_idc									
	0	1	2	3	4	5	6	7	8	9
initValue	95	79	63	31	31	95	79	63	31	31

Table 9-18 – Values of initValue for ctxIdx of ref_idx_l0 and ref_idx_l1

Initialization variable	ctxIdx of ref_idx_l0 and ref_idx_l1			
	0	1	2	3
initValue	153	153	153	153

Table 9-19 – Values of initValue for ctxIdx of mvp_l0_flag and mvp_l1_flag

Initialization variable	ctxIdx of mvp_l0_flag and mvp_l1_flag	
	0	1
initValue	168	168

Table 9-20 – Values of initValue for ctxIdx of split_transform_flag

Initialization variable	ctxIdx of split_transform_flag								
	0	1	2	3	4	5	6	7	8
initValue	153	138	138	124	138	94	224	167	122

Table 9-21 – Values of initValue for ctxIdx of cbf_luma

Initialization variable	ctxIdx of cbf_luma					
	0	1	2	3	4	5
initValue	111	141	153	111	153	111

Table 9-22 – Values of initValue for ctxIdx of cbf_cb and cbf_cr

Initialization variable	ctxIdx of cbf_cb and cbf_cr														
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
initValue	94	138	182	154	149	107	167	154	149	92	167	154	154	154	154

Table 9-23 – Values of initValue for ctxIdx of abs_mvd_greater0_flag and abs_mvd_greater1_flag

Initialization variable	ctxIdx of abs_mvd_greater0_flag and abs_mvd_greater1_flag			
	0	1	2	3
initValue	140	198	169	198

Table 9-24 – Values of initValue for ctxIdx of cu_qp_delta_abs

Initialization variable	ctxIdx of cu_qp_delta_abs					
	0	1	2	3	4	5
initValue	154	154	154	154	154	154

Table 9-25 – Values of initValue for ctxIdx of transform_skip_flag

Initialization variable	ctxIdx of transform_skip_flag					
	0	1	2	3	4	5
initValue	139	139	139	139	139	139

Table 9-26 – Values of initValue for ctxIdx of last_sig_coeff_x_prefix

Initialization variable	ctxIdx of last_sig_coeff_x_prefix																	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
initValue	110	110	124	125	140	153	125	127	140	109	111	143	127	111	79	108	123	63
	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
initValue	125	110	94	110	95	79	125	111	110	78	110	111	111	95	94	108	123	108
	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53
initValue	125	110	124	110	95	94	125	111	111	79	125	126	111	111	79	108	123	93

Table 9-27 – Values of initValue for ctxIdx of last_sig_coeff_y_prefix

Initialization variable	ctxIdx of last_sig_coeff_y_prefix																	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
initValue	110	110	124	125	140	153	125	127	140	109	111	143	127	111	79	108	123	63
	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
initValue	125	110	94	110	95	79	125	111	110	78	110	111	111	95	94	108	123	108
	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53
initValue	125	110	124	110	95	94	125	111	111	79	125	126	111	111	79	108	123	93

Table 9-28 – Values of initValue for ctxIdx of coded_sub_block_flag

Initialization variable	ctxIdx of coded_sub_block_flag											
	0	1	2	3	4	5	6	7	8	9	10	11
initValue	91	171	134	141	121	140	61	154	121	140	61	154

Table 9-29 – Values of initValue for ctxIdx of sig_coeff_flag

Initialization variable	ctxIdx of sig_coeff_flag															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
initValue	111	111	125	110	110	94	124	108	124	107	125	141	179	153	125	107
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
initValue	125	141	179	153	125	107	125	141	179	153	125	140	139	182	182	152
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
initValue	136	152	136	153	136	139	111	136	139	111	155	154	139	153	139	123
	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
initValue	123	63	153	166	183	140	136	153	154	166	183	140	136	153	154	166
	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
initValue	183	140	136	153	154	170	153	123	123	107	121	107	121	167	151	183
	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
initValue	140	151	183	140	170	154	139	153	139	123	123	63	124	166	183	140
	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
initValue	136	153	154	166	183	140	136	153	154	166	183	140	136	153	154	170
	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
initValue	153	138	138	122	121	122	121	167	151	183	140	151	183	140	141	111
	128	129	130	131												
initValue	140	140	140	140												

Table 9-30 – Values of initValue for ctxIdx of coeff_abs_level_greater1_flag

Initialization variable	ctxIdx of coeff_abs_level_greater1_flag															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
initValue	140	92	137	138	140	152	138	139	153	74	149	92	139	107	122	152
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
initValue	140	179	166	182	140	227	122	197	154	196	196	167	154	152	167	182
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
initValue	182	134	149	136	153	121	136	137	169	194	166	167	154	167	137	182
	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
initValue	154	196	167	167	154	152	167	182	182	134	149	136	153	121	136	122
	64	65	66	67	68	69	70	71								
initValue	169	208	166	167	154	152	167	182								

Table 9-31 – Values of initValue for ctxIdx of coeff_abs_level_greater2_flag

Initialization variable	ctxIdx of coeff_abs_level_greater2_flag								
	0	1	2	3	4	5	6	7	8
initValue	138	153	136	167	152	152	107	167	91
	9	10	11	12	13	14	15	16	17
initValue	122	107	167	107	167	91	107	107	167

Table 9-32 – Values of initValue for ctxIdx of explicit_rdpem_flag

Initialization variable	ctxIdx of explicit_rdpem_flag			
	0	1	2	3
initValue	139	139	139	139

Table 9-33 – Values of initValue for ctxIdx of explicit_rdpem_dir_flag

Initialization variable	ctxIdx of explicit_rdpem_dir_flag			
	0	1	2	3
initValue	139	139	139	139

Table 9-34 – Values of initValue for ctxIdx of cu_chroma_qp_offset_flag

Initialization variable	ctxIdx of cu_chroma_qp_offset_flag		
	0	1	2
initValue	154	154	154

Table 9-35 – Values of initValue for ctxIdx of cu_chroma_qp_offset_idx

Initialization variable	ctxIdx of cu_chroma_qp_offset_idx		
	0	1	2
initValue	154	154	154

Table 9-36 – Values of initValue for ctxIdx of log2_res_scale_abs_plus1

Initialization variable	ctxIdx of log2_res_scale_abs_plus1								
	0	1	2	3	4	5	6	7	8
initValue	154	154	154	154	154	154	154	154	154
	9	10	11	12	13	14	15	16	17
initValue	154	154	154	154	154	154	154	154	154
	18	19	20	21	22	23			
initValue	154	154	154	154	154	154			

Table 9-37 – Values of initValue for ctxIdx of res_scale_sign_flag

Initialization variable	ctxIdx of res_scale_sign_flag					
	0	1	2	3	4	5
initValue	154	154	154	154	154	154

Table 9-38 – Values of initValue for ctxIdx of palette_mode_flag

Initialization variable	ctxIdx of palette_mode_flag		
	0	1	2
initValue	154	154	154

Table 9-39 – Values of initValue for ctxIdx of tu_residual_act_flag

Initialization variable	ctxIdx of tu_residual_act_flag		
	0	1	2
initValue	154	154	154

Table 9-40 – Values of initValue for ctxIdx of palette_run_prefix

Initialization variable	ctxIdx of palette_run_prefix											
	0	1	2	3	4	5	6	7	8	9	10	11
initValue	154	154	154	154	154	154	154	154	154	154	154	154
	12	13	14	15	16	17	18	19	20	21	22	23
initValue	154	154	154	154	154	154	154	154	154	154	154	154

Table 9-41 – Values of initValue for ctxIdx of copy_above_palette_indices_flag and copy_above_indices_for_final_run_flag

Initialization variable	ctxIdx of copy_above_palette_indices_flag and copy_above_indices_for_final_run_flag		
	0	1	2
initValue	154	154	154

Table 9-42 – Values of initValue for ctxIdx of palette_transpose_flag

Initialization variable	ctxIdx of palette_transpose_flag		
	0	1	2
initValue	154	154	154

9.3.2.3 Initialization process for palette predictor entries

Outputs of this process are the initialized palette predictor variables PredictorPaletteSize and PredictorPaletteEntries.

The variable numComps is derived as follows:

$$\text{numComps} = (\text{ChromaArrayType} == 0) ? 1 : 3 \quad (9-8)$$

- If pps_palette_predictor_initializers_present_flag is equal to 1, the following applies:

- PredictorPaletteSize is set equal to pps_num_palette_predictor_initializers.
- The array PredictorPaletteEntries is derived as follows:

$$\begin{aligned} &\text{for(comp = 0; comp < numComps; comp++)} \\ &\quad \text{for(i = 0; i < PredictorPaletteSize; i++)} \\ &\quad \quad \text{PredictorPaletteEntries[comp][i] = pps_palette_predictor_initializer[comp][i]} \end{aligned} \quad (9-9)$$

- Otherwise (pps_palette_predictor_initializers_present_flag is equal to 0), if sps_palette_predictor_initializers_present_flag is equal to 1, the following applies:
- PredictorPaletteSize is set equal to sps_num_palette_predictor_initializers_minus1 plus 1.
- The array PredictorPaletteEntries is derived as follows:

$$\begin{aligned} &\text{for(comp = 0; comp < numComps; comp++)} \\ &\quad \text{for(i = 0; i < PredictorPaletteSize; i++)} \\ &\quad \quad \text{PredictorPaletteEntries[comp][i] = sps_palette_predictor_initializer[comp][i]} \end{aligned} \quad (9-10)$$

Otherwise (pps_palette_predictor_initializers_present_flag is equal to 0 and sps_palette_predictor_initializers_present_flag is equal to 0), PredictorPaletteSize is set equal to 0.

9.3.2.4 Storage process for context variables, Rice parameter initialization states, and palette predictor variables

Inputs to this process are:

- The CABAC context variables indexed by ctxTable and ctxIdx.
- The Rice parameter initialization states indexed by k.
- The palette predictor variables, PredictorPaletteSize and PredictorPaletteEntries.

Outputs of this process are:

- The variables tableStateSync and tableMPSSync containing the values of the variables pStateIdx and valMps used in the initialization process of context variables and Rice parameter initialization states that are assigned to all syntax elements in clauses 7.3.8.1 through 7.3.8.12, except end_of_slice_segment_flag, end_of_subset_one_bit and pcm_flag.

- The variables tableStatCoeffSync containing the values of the variables StatCoeff[k] used in the initialization process of context variables and Rice parameter initialization states.
- The variables PredictorPaletteSizeSync and tablePredictorPaletteEntriesSync containing the values used in the initialization process of palette predictor variables.

For each context variable, the corresponding entries pStateIdx and valMps of tables tableStateSync and tableMPSSync are initialized to the corresponding pStateIdx and valMps.

For each Rice parameter initialization state k, each entry of the table tableStatCoeffSync is initialized to the corresponding value of StatCoeff[k].

For palette predictor variables, PredictorPaletteSizeSync is initialized to PredictorPaletteSize. For tablePredictorPaletteEntriesSync, each entry is initialized to the corresponding value of PredictorPaletteEntries.

The storage process for context variables is illustrated in the flowchart of Figure 9-4.

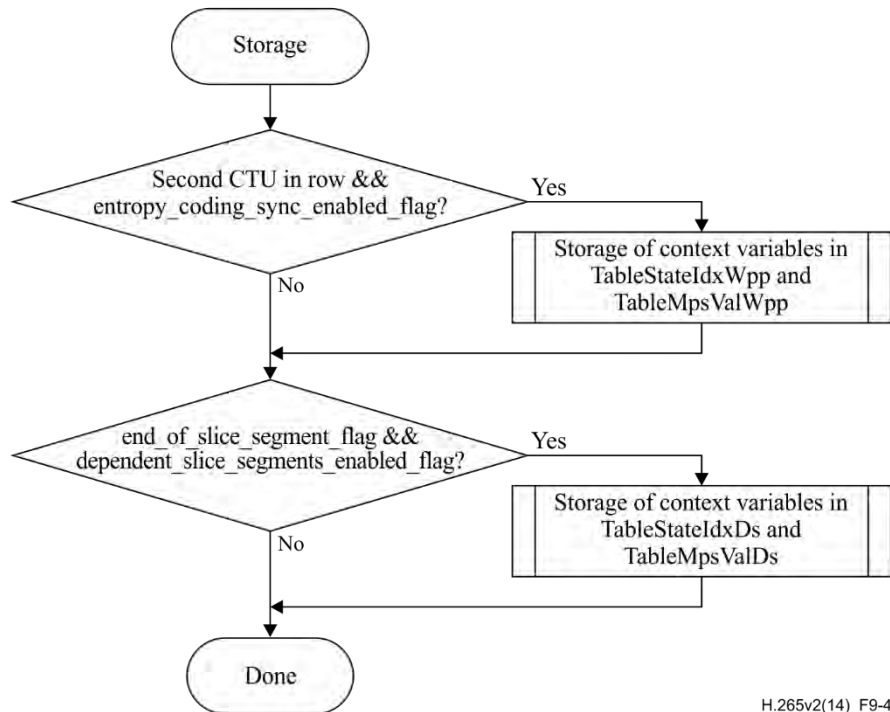


Figure 9-4 – Illustration of CABAC storage process (informative)

9.3.2.5 Synchronization process for context variables, Rice parameter initialization states, and palette predictor variables

Inputs to this process are:

- The variables tableStateSync and tableMPSSync containing the values of the variables pStateIdx and valMps used in the storage process of context variables that are assigned to all syntax elements in clauses 7.3.8.1 through 7.3.8.12, except end_of_slice_segment_flag, end_of_subset_one_bit and pcm_flag.
- The variable tableStatCoeffSync containing the values of the variables StatCoeff[k] used in the storage process of context variables and Rice parameter initialization states.
- The variables PredictorPaletteSizeSync and tablePredictorPaletteEntriesSync containing the values used in the storage process of palette predictor variables.

Outputs of this process are:

- The initialized CABAC context variables indexed by ctxTable and ctxIdx.
- The initialized Rice parameter initialization states StatCoeff indexed by k.
- The palette predictor variables, PredictorPaletteSize and PredictorPaletteEntries.

For each context variable, the corresponding context variables pStateIdx and valMps are initialized to the corresponding entries pStateIdx and valMps of tables tableStateSync and tableMPSSync.

For each Rice parameter initialization state, each variable StatCoeff[k] is initialized to the corresponding entry of tableStatCoeffSync.

For palette predictor variables, PredictorPaletteSize is initialized to PredictorPaletteSizeSync. For PredictorPaletteEntries, each entry is initialized to the corresponding value of tablePredictorPaletteEntriesSync.

9.3.2.6 Initialization process for the arithmetic decoding engine

Outputs of this process are the initialized decoding engine registers ivlCurrRange and ivlOffset both in 16 bit register precision.

The status of the arithmetic decoding engine is represented by the variables ivlCurrRange and ivlOffset. In the initialization procedure of the arithmetic decoding process, ivlCurrRange is set equal to 510 and ivlOffset is set equal to the value returned from read_bits(9) interpreted as a 9 bit binary representation of an unsigned integer with the most significant bit written first.

The bitstream shall not contain data that result in a value of ivlOffset being equal to 510 or 511.

NOTE – The description of the arithmetic decoding engine in this Specification utilizes 16 bit register precision. However, a minimum register precision of 9 bits is required for storing the values of the variables ivlCurrRange and ivlOffset after invocation of the arithmetic decoding process (DecodeBin) as specified in clause 9.3.4.3. The arithmetic decoding process for a binary decision (DecodeDecision) as specified in clause 9.3.4.3.2 and the decoding process for a binary decision before termination (DecodeTerminate) as specified in clause 9.3.4.3.5 require a minimum register precision of 9 bits for the variables ivlCurrRange and ivlOffset. The bypass decoding process for binary decisions (DecodeBypass) as specified in clause 9.3.4.3.4 requires a minimum register precision of 10 bits for the variable ivlOffset and a minimum register precision of 9 bits for the variable ivlCurrRange.

9.3.3 Binarization process

9.3.3.1 General

Input to this process is a request for a syntax element.

Output of this process is the binarization of the syntax element.

Table 9-43 specifies the type of binarization process associated with each syntax element and corresponding inputs.

The specification of the truncated Rice (TR) binarization process, the k-th order Exp-Golomb (EGk) binarization process, limited k-th order Exp-Golomb (EGk) binarization process, the fixed-length (FL) binarization process, and the truncated binary binarization process are given in clauses 9.3.3.2 through 9.3.3.6, respectively. Other binarizations are specified in clauses 9.3.3.7 through 9.3.3.14.

Table 9-43 – Syntax elements and associated binarizations

Syntax structure	Syntax element	Binarization	
		Process	Input parameters
slice_segment_data()	end_of_slice_segment_flag	FL	cMax = 1
	end_of_subset_one_bit	FL	cMax = 1
sao()	sao_merge_left_flag	FL	cMax = 1
	sao_merge_up_flag	FL	cMax = 1
	sao_type_idx_luma	TR	cMax = 2, cRiceParam = 0
	sao_type_idx_chroma	TR	cMax = 2, cRiceParam = 0
	sao_offset_abs[][][]	TR	cMax = (1 << (Min(bitDepth, 10) - 5)) - 1, cRiceParam = 0
	sao_offset_sign[][][]	FL	cMax = 1
	sao_band_position[][][]	FL	cMax = 31
	sao_eo_class_luma	FL	cMax = 3
	sao_eo_class_chroma	FL	cMax = 3

Table 9-43 – Syntax elements and associated binarizations

Syntax structure	Syntax element	Binarization	
		Process	Input parameters
coding_quadtree()	split_cu_flag[][]	FL	cMax = 1
coding_unit()	cu_transquant_bypass_flag	FL	cMax = 1
	cu_skip_flag	FL	cMax = 1
	palette_mode_flag	FL	cMax = 1
	pred_mode_flag	FL	cMax = 1
	part_mode	9.3.3.7	(xCb, yCb) = (x0, y0), log2CbSize
	pcm_flag[][]	FL	cMax = 1
	prev_intra_luma_pred_flag[][]	FL	cMax = 1
	mpm_idx[][]	TR	cMax = 2, cRiceParam = 0
	rem_intra_luma_pred_mode[][]	FL	cMax = 31
	intra_chroma_pred_mode[][]	9.3.3.8	-
	rqt_root_cbf	FL	cMax = 1
prediction_unit()	merge_flag[][]	FL	cMax = 1
	merge_idx[][]	TR	cMax = MaxNumMergeCand – 1, cRiceParam = 0
	inter_pred_idc[x0][y0]	9.3.3.9	nPbW, nPbH
	ref_idx_l0[][]	TR	cMax = num_ref_idx_l0_active_minus1, cRiceParam = 0
	mvp_l0_flag[][]	FL	cMax = 1
	ref_idx_l1[][]	TR	cMax = num_ref_idx_l1_active_minus1, cRiceParam = 0
	mvp_l1_flag[][]	FL	cMax = 1
transform_tree()	split_transform_flag[][][]	FL	cMax = 1
	cbf_luma[][][]	FL	cMax = 1
	cbf_cb[][][]	FL	cMax = 1
	cbf_cr[][][]	FL	cMax = 1
mvd_coding()	abs_mvd_greater0_flag[]	FL	cMax = 1
	abs_mvd_greater1_flag[]	FL	cMax = 1
	abs_mvd_minus2[]	EG1	-
	mvd_sign_flag[]	FL	cMax = 1
transform_unit()	tu_residual_act_flag	FL	cMax = 1

Table 9-43 – Syntax elements and associated binarizations

Syntax structure	Syntax element	Binarization	
		Process	Input parameters
cross_comp_pred()	log2_res_scale_abs_plus1	TR	cMax = 4, cRiceParam = 0
	res_scale_sign_flag	FL	cMax = 1
residual_coding()	transform_skip_flag[][]	FL	cMax = 1
	explicit_rdpem_flag[][]	FL	cMax = 1
	explicit_rdpem_dir_flag[][]	FL	cMax = 1
	last_sig_coeff_x_prefix	TR	cMax = (log2TrafoSize << 1) – 1, cRiceParam = 0
	last_sig_coeff_y_prefix	TR	cMax = (log2TrafoSize << 1) – 1, cRiceParam = 0
	last_sig_coeff_x_suffix	FL	cMax = (1 << ((last_sig_coeff_x_prefix >> 1) – 1) – 1)
	last_sig_coeff_y_suffix	FL	cMax = (1 << ((last_sig_coeff_y_prefix >> 1) – 1) – 1)
	coded_sub_block_flag[][]	FL	cMax = 1
	sig_coeff_flag[][]	FL	cMax = 1
	coeff_abs_level_greater1_flag[]	FL	cMax = 1
	coeff_abs_level_greater2_flag[]	FL	cMax = 1
	coeff_abs_level_remaining[]	9.3.3.11	current sub-block scan index i, baseLevel
	coeff_sign_flag[]	FL	cMax = 1
palette_coding()	palette_predictor_run	EG0	-
	num_signalled_palette_entries	EG0	-
	new_palette_entries	FL	cMax = cIdx == 0 ? ((1 << BitDepth _V) – 1) : ((1 << BitDepth _C) – 1)
	palette_escape_val_present_flag	FL	cMax = 1
	num_palette_indices_minus1	9.3.3.14	MaxPaletteIndex
	palette_idx_idc	9.3.3.13	MaxPaletteIndex
	copy_above_indices_for_final_run_flag	FL	cMax = 1
	palette_transpose_flag	FL	cMax = 1
	copy_above_palette_indices_flag	FL	cMax = 1
	palette_run_prefix	TR	cMax = Floor(Log2(PaletteMaxRunMinus1)) + 1, cRiceParam = 0
	palette_run_suffix	TB	cMax = (PrefixOffset << 1) > PaletteMaxRunMinus1 ? (PaletteMaxRunMinus1 – PrefixOffset) : (PrefixOffset – 1)
delta_qp()	cu_qp_delta_abs	9.3.3.10	-
	cu_qp_delta_sign_flag	FL	cMax = 1
chroma_qp_offset()	cu_chroma_qp_offset_flag	FL	cMax = 1
	cu_chroma_qp_offset_idx	TR	cMax = chroma_qp_offset_list_len_minus1, cRiceParam = 0

9.3.3.2 Truncated Rice binarization process

Input to this process is a request for a truncated Rice (TR) binarization, cMax and cRiceParam.

Output of this process is the TR binarization associating each value symbolVal with a corresponding bin string.

A TR bin string is a concatenation of a prefix bin string and, when present, a suffix bin string.

For the derivation of the prefix bin string, the following applies:

- The prefix value of symbolVal, prefixVal, is derived as follows:

$$\text{prefixVal} = \text{symbolVal} \gg \text{cRiceParam} \quad (9-11)$$

- The prefix of the TR bin string is specified as follows:
 - If prefixVal is less than cMax >> cRiceParam, the prefix bin string is a bit string of length prefixVal + 1 indexed by binIdx. The bins for binIdx less than prefixVal are equal to 1. The bin with binIdx equal to prefixVal is equal to 0. Table 9-44 illustrates the bin strings of this unary binarization for prefixVal.
 - Otherwise, the bin string is a bit string of length cMax >> cRiceParam with all bins being equal to 1.

Table 9-44 – Bin string of the unary binarization (informative)

prefixVal	Bin string					
0	0					
1	1	0				
2	1	1	0			
3	1	1	1	0		
4	1	1	1	1	0	
5	1	1	1	1	1	0
...						
binIdx	0	1	2	3	4	5

When cMax is greater than symbolVal and cRiceParam is greater than 0, the suffix of the TR bin string is present and it is derived as follows:

- The suffix value suffixVal is derived as follows:

$$\text{suffixVal} = \text{symbolVal} - ((\text{prefixVal}) \ll \text{cRiceParam}) \quad (9-12)$$

- The suffix of the TR bin string is specified by invoking the fixed-length (FL) binarization process as specified in clause 9.3.3.5 for suffixVal with a cMax value equal to $(1 \ll \text{cRiceParam}) - 1$.

NOTE – For the input parameter cRiceParam = 0, the TR binarization is exactly a truncated unary binarization and it is always invoked with a cMax value equal to the largest possible value of the syntax element being decoded.

9.3.3.3 k-th order Exp-Golomb binarization process

Inputs to this process is a request for a k-th order Exp-Golomb (EGk) binarization.

Output of this process is the EGk binarization associating each value symbolVal with a corresponding bin string.

The bin string of the EGk binarization process for each value symbolVal is specified as follows, where each call of the function put(X), with X being equal to 0 or 1, adds the binary value X at the end of the bin string:

```

absV = Abs( symbolVal )
stopLoop = 0
do

```

```

    if( absV >= ( 1 << k ) ) {
        put( 1 )
        absV = absV - ( 1 << k )
        k++
    } else {
        put( 0 )
        while( k-- )
            put( ( absV >> k ) & 1 )
        stopLoop = 1
    }
while( !stopLoop )

```

(9-13)

NOTE – The specification for the k-th order Exp-Golomb (EGk) code uses 1's and 0's in reverse meaning for the unary part of the Exp-Golomb code of 0-th order as specified in clause 9.2.

9.3.3.4 Limited EGk binarization process

This process is only invoked when `extended_precision_processing_flag` is equal to 1.

Inputs to this process is a request for a limited EGk binarization, the Rice parameter `riceParam` and the colour component `cIdx`.

Output of this process is the limited EGk binarization associating each value `symbolVal` with a corresponding bin string.

The variables `log2TransformRange` and `maxPrefixExtensionLength` are derived as follows:

$$\text{log2TransformRange} = \text{cIdx} == 0 ? \text{Max}(15, \text{BitDepth}_Y + 6) : \text{Max}(15, \text{BitDepth}_C + 6) \quad (9-14)$$

$$\text{maxPrefixExtensionLength} = 28 - \text{log2TransformRange} \quad (9-15)$$

The bin string of the limited EGk binarization process for each value `symbolVal` is specified as follows, where each call of the function `put(X)`, with `X` being equal to 0 or 1, adds the binary value `X` at the end of the bin string:

```

codeValue = symbolVal >> riceParam
PrefixExtensionLength = 0
while( ( PrefixExtensionLength < maxPrefixExtensionLength ) &&
      ( codeValue > ( ( 2 << PrefixExtensionLength ) - 2 ) ) ) {
    PrefixExtensionLength++
    put( 1 )
}
if( PrefixExtensionLength == maxPrefixExtensionLength )
    escapeLength = log2TransformRange
else {
    escapeLength = PrefixExtensionLength + riceParam
    put( 0 )
}
symbolVal = symbolVal - ( ( ( 1 << PrefixExtensionLength ) - 1 ) << riceParam )
while( ( escapeLength-- ) > 0 )
    put( ( symbolVal >> escapeLength ) & 1 )

```

(9-16)

9.3.3.5 Fixed-length binarization process

Inputs to this process are a request for a fixed-length (FL) binarization and `cMax`.

Output of this process is the FL binarization associating each value `symbolVal` with a corresponding bin string.

FL binarization is constructed by using the `fixedLength`-bit unsigned integer bin string of the symbol value `symbolVal`, where `fixedLength = Ceil(Log2(cMax + 1))`. The indexing of bins for the FL binarization is such that the `binIdx = 0` relates to the most significant bit with increasing values of `binIdx` towards the least significant bit.

9.3.3.6 Truncated Binary (TB) binarization process

Input to this process is a request for a TB binarization for a syntax element with value `synVal` and `cMax`. Output of this process is the TB binarization of the syntax element. The bin string of the TB binarization process of a syntax element `synVal` is specified as follows:

$$\begin{aligned} n &= cMax + 1 \\ k &= \text{Floor}(\text{Log}_2(n)) \\ u &= (1 \ll (k + 1)) - n \end{aligned} \quad (9-17)$$

- If `synVal` is less than `u`, the TB bin string is derived by invoking the FL binarization process specified in clause 9.3.3.5 for `synVal` with a `cMax` value equal to $(1 \ll k) - 1$.
- Otherwise (`synVal` is greater than or equal to `u`), the TB bin string is derived by invoking the FL binarization process specified in clause 9.3.3.5 for $(\text{synVal} + u)$ with a `cMax` value equal to $(1 \ll (k + 1)) - 1$.

9.3.3.7 Binarization process for `part_mode`

Inputs to this process are a request for a binarization for the syntax element `part_mode`, a luma location (x_{Cb}, y_{Cb}) , specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture and a variable `log2CbSize` specifying the current luma coding block size.

Output of this process is the binarization of the syntax element.

The binarization for the syntax element `part_mode` is specified in Table 9-45 depending on the values of `CuPredMode[xCb][yCb]` and `log2CbSize`.

Table 9-45 – Binarization for `part_mode`

CuPredMode[xCb][yCb]	part_mode	PartMode	Bin string			
			log2CbSize > MinCbLog2SizeY		log2CbSize == MinCbLog2SizeY	
			!amp_enabled_flag	amp_enabled_flag	log2CbSize == 3	log2CbSize > 3
MODE_INTRA	0	PART_2Nx2N	-	-	1	1
	1	PART_NxN	-	-	0	0
MODE_INTER	0	PART_2Nx2N	1	1	1	1
	1	PART_2NxN	01	011	01	01
	2	PART_Nx2N	00	001	00	001
	3	PART_NxN	-	-	-	000
	4	PART_2NxN	-	0100	-	-
	5	PART_2NxND	-	0101	-	-
	6	PART_nLx2N	-	0000	-	-
	7	PART_nRx2N	-	0001	-	-

9.3.3.8 Binarization process for `intra_chroma_pred_mode`

Input to this process is a request for a binarization for the syntax element `intra_chroma_pred_mode`.

Output of this process is the binarization of the syntax element.

The binarization for the syntax element `intra_chroma_pred_mode` is specified in Table 9-46.

Table 9-46 – Binarization for intra_chroma_pred_mode

Value of intra_chroma_pred_mode	Bin string
4	0
0	100
1	101
2	110
3	111

9.3.3.9 Binarization process for inter_pred_idc

Inputs to this process are a request for a binarization for the syntax element `inter_pred_idc`, the current luma prediction block width `nPbW` and the current luma prediction block height `nPbH`.

Output of this process is the binarization of the syntax element.

The binarization for the syntax element `inter_pred_idc` is specified in Table 9-47.

Table 9-47 – Binarization for inter_pred_idc

Value of inter_pred_idc	Name of inter_pred_idc	Bin string	
		(nPbW + nPbH) != 12	(nPbW + nPbH) == 12
0	PRED_L0	00	0
1	PRED_L1	01	1
2	PRED_BI	1	-

9.3.3.10 Binarization process for cu_qp_delta_abs

Input to this process is a request for a binarization for the syntax element `cu_qp_delta_abs`.

Output of this process is the binarization of the syntax element.

The binarization of the syntax element `cu_qp_delta_abs` is a concatenation of a prefix bin string and (when present) a suffix bin string.

For the derivation of the prefix bin string, the following applies:

- The prefix value of `cu_qp_delta_abs`, `prefixVal`, is derived as follows:

$$\text{prefixVal} = \text{Min}(\text{cu_qp_delta_abs}, 5) \quad (9-18)$$

- The prefix bin string is specified by invoking the TR binarization process as specified in clause 9.3.3.2 for `prefixVal` with `cMax = 5` and `cRiceParam = 0`.

When `prefixVal` is greater than 4, the suffix bin string is present and it is derived as follows:

- The suffix value of `cu_qp_delta_abs`, `suffixVal`, is derived as follows:

$$\text{suffixVal} = \text{cu_qp_delta_abs} - 5 \quad (9-19)$$

- The suffix bin string is specified by invoking the k-th order EGk binarization process as specified in clause 9.3.3.3 for `suffixVal` with the Exp-Golomb order `k` set equal to 0.

9.3.3.11 Binarization process for coeff_abs_level_remaining[]

Input to this process is a request for a binarization for the syntax element `coeff_abs_level_remaining[n]`, the current sub-block scan index `i`, `baseLevel`, the colour component `cIdx` and the luma location (`x0`, `y0`) specifying the top-left sample of the current luma transform block relative to the top-left luma sample of the picture.

Output of this process is the binarization of the syntax element.

Depending on the value of `persistent_rice_adaptation_enabled_flag`, the following applies:

- If `persistent_rice_adaptation_enabled_flag` is equal to 0, the variable `initRiceValue` is set equal to 0.
- Otherwise (`persistent_rice_adaptation_enabled_flag` is equal to 1), the following applies:
 - The variable `sbType` is derived as follows:

- If `transform_skip_flag[x0][y0][cIdx]` is equal to 0 and `cu_transquant_bypass_flag` is equal to 0, the following applies:

$$sbType = 2 * (cIdx == 0 ? 1 : 0) \quad (9-20)$$

- Otherwise, the following applies:

$$sbType = 2 * (cIdx == 0 ? 1 : 0) + 1 \quad (9-21)$$

- The variable `initRiceValue` is derived as follows:

$$initRiceValue = StatCoeff[sbType] / 4 \quad (9-22)$$

- If this process is invoked for the first time for the current sub-block scan index `i`, `StatCoeff[sbType]` is modified as follows:

$$\begin{aligned} & \text{if(coeff_abs_level_remaining[n] } \geq (3 \ll (StatCoeff[sbType] / 4))) \\ & \quad StatCoeff[sbType]++ \\ & \text{else if(} 2 * \text{coeff_abs_level_remaining[n]} < (1 \ll (StatCoeff[sbType] / 4)) \ \&\& \\ & \quad StatCoeff[sbType] > 0) \\ & \quad StatCoeff[sbType]-- \end{aligned} \quad (9-23)$$

The variables `cLastAbsLevel` and `cLastRiceParam` are derived as follows:

- If this process is invoked for the first time for the current sub-block scan index `i`, `cLastAbsLevel` is set equal to 0 and `cLastRiceParam` is set equal to `initRiceValue`.
- Otherwise (this process is not invoked for the first time for the current sub-block scan index `i`), `cLastAbsLevel` and `cLastRiceParam` are set equal to the values of `cAbsLevel` and `cRiceParam`, respectively, that have been derived during the last invocation of the binarization process for the syntax element `coeff_abs_level_remaining[n]` as specified in this clause.

The variable `cAbsLevel` is set equal to `baseLevel + coeff_abs_level_remaining[n]`.

The variable `cRiceParam` is derived from `cLastAbsLevel` and `cLastRiceParam` as follows:

- If `persistent_rice_adaptation_enabled_flag` is equal to 0, the following applies:

$$\begin{aligned} cRiceParam = \\ \text{Min(cLastRiceParam + (cLastAbsLevel } > (3 * (1 \ll cLastRiceParam)) ? 1 : 0), 4) \end{aligned} \quad (9-24)$$

- Otherwise (`persistent_rice_adaptation_enabled_flag` is equal to 1), the following applies:

$$cRiceParam = cLastRiceParam + (cLastAbsLevel > (3 * (1 \ll cLastRiceParam)) ? 1 : 0) \quad (9-25)$$

The variable `cMax` is derived from `cRiceParam` as:

$$cMax = 4 \ll cRiceParam \quad (9-26)$$

The binarization of the syntax element `coeff_abs_level_remaining[n]` is a concatenation of a prefix bin string and (when present) a suffix bin string.

For the derivation of the prefix bin string, the following applies:

- The prefix value of `coeff_abs_level_remaining[n]`, `prefixVal`, is derived as follows:

$$prefixVal = \text{Min(cMax, coeff_abs_level_remaining[n])} \quad (9-27)$$

- The prefix bin string is specified by invoking the TR binarization process as specified in clause 9.3.3.2 for prefixVal with the variables cMax and cRiceParam as inputs.

When the prefix bin string is equal to the bit string of length 4 with all bits equal to 1, the suffix bin string is present and it is derived as follows:

- The suffix value of coeff_abs_level_remaining[n], suffixVal, is derived as follows:

$$\text{suffixVal} = \text{coeff_abs_level_remaining}[n] - \text{cMax} \quad (9-28)$$

- If extended_precision_processing_flag is equal to 0, the suffix bin string is specified by invoking the k-th order EGk binarization process as specified in clause 9.3.3.3 for the binarization of suffixVal with the Exp-Golomb order k set equal to cRiceParam + 1.
- Otherwise (extended_precision_processing_flag is equal to 1), the suffix bin string is specified by invoking the limited k-th order EGk binarization process as specified in clause 9.3.3.4 for the binarization of suffixVal with the variable riceParam set equal to cRiceParam + 1 and the colour component cIdx.

9.3.3.12 Binarization process for palette_escape_val

Input to this process is a request for a binarization for the syntax element palette_escape_val, cu_transquant_bypass_flag and colour component index cIdx.

Output of this process is the binarization of palette_escape_val.

The variable bitDepth is derived as follows:

$$\text{bitDepth} = (\text{cIdx} == 0) ? \text{BitDepth}_Y : \text{BitDepth}_C \quad (9-29)$$

The binarization of palette_escape_val is derived as follows:

- If cu_transquant_bypass_flag is equal to 1, the binarization of palette_escape_val is derived by invoking the FL binarization process specified in clause 9.3.3.5 with the input parameter set to $(1 \ll \text{bitdepth}) - 1$.

Otherwise (cu_transquant_bypass_flag is equal to 0), the binarization of palette_escape_val is derived by invoking the k-th order Exp-Golomb binarization process specified in clause 9.3.3.3 with k set equal to 3.

9.3.3.13 Binarization process for palette_idx_idc

Input to this process is a request for a binarization for the syntax element palette_idx_idc and the variable MaxPaletteIndex.

Output of this process is the binarization of the syntax element.

The variable cMax is derived as follows:

- If this process is invoked for the first time for the current block, cMax is set equal to MaxPaletteIndex.
- Otherwise (this process is not invoked for the first time for the current block), cMax is set equal to MaxPaletteIndex minus 1.

The binarization for the palette_idx_idc is derived by invoking the TB binarization process specified in clause 9.3.3.6 with cMax.

9.3.3.14 Binarization process for num_palette_indices_minus1

Input to this process is a request for a binarization for the syntax element num_palette_indices_minus1, and MaxPaletteIndex.

Output of this process is the binarization of the syntax element.

The variable cRiceParam is derived as follows:

$$\text{cRiceParam} = 3 + ((\text{MaxPaletteIndex} + 1) \gg 3) \quad (9-30)$$

The variable cMax is derived from cRiceParam as:

$$\text{cMax} = 4 \ll \text{cRiceParam} \quad (9-31)$$

The binarization of the syntax element num_palette_indices_minus1 is a concatenation of a prefix bin string and (when present) a suffix bin string.

For the derivation of the prefix bin string, the following applies:

- The prefix value of `num_palette_indices_minus1`, `prefixVal`, is derived as follows:

$$\text{prefixVal} = \text{Min}(\text{cMax}, \text{num_palette_indices_minus1}) \quad (9-32)$$

- The prefix bin string is specified by invoking the TR binarization process as specified in clause 9.3.3.2 for `prefixVal` with the variables `cMax` and `cRiceParam` as inputs.

When the prefix bin string is equal to the bit string of length 4 with all bits equal to 1, the suffix bin string is present and it is derived as follows:

- The suffix value of `num_palette_indices_minus1`, `suffixVal`, is derived as follows:

$$\text{suffixVal} = \text{num_palette_indices_minus1} - \text{cMax} \quad (9-33)$$

The suffix bin string is specified by invoking the k-th order EGk binarization process as specified in clause 9.3.3.3 for the binarization of `suffixVal` with the Exp-Golomb order `k` set equal to `cRiceParam + 1`.

9.3.4 Decoding process flow

9.3.4.1 General

Inputs to this process are all bin strings of the binarization of the requested syntax element as specified in clause 9.3.3.

Output of this process is the value of the syntax element.

This process specifies how each bin of a bin string is parsed for each syntax element. After parsing each bin, the resulting bin string is compared to all bin strings of the binarization of the syntax element and the following applies:

- If the bin string is equal to one of the bin strings, the corresponding value of the syntax element is the output.
- Otherwise (the bin string is not equal to one of the bin strings), the next bit is parsed.

While parsing each bin, the variable `binIdx` is incremented by 1 starting with `binIdx` being set equal to 0 for the first bin.

The parsing of each bin is performed by invoking the derivation process for `ctxTable`, `ctxIdx`, and `bypassFlag` as specified in clause 9.3.4.2 with `binIdx` as input and `ctxTable`, `ctxIdx` and `bypassFlag` as outputs.

NOTE – As a consequence of invoking the process specified in clause 9.3.4.2, the arithmetic decoding process as specified in clause 9.3.4.3 is invoked with `ctxTable`, `ctxIdx` and `bypassFlag` as inputs and the value of the bin as output.

9.3.4.2 Derivation process for `ctxTable`, `ctxIdx` and `bypassFlag`

9.3.4.2.1 General

Input to this process is the position of the current bin within the bin string, `binIdx`.

Outputs of this process are `ctxTable`, `ctxIdx` and `bypassFlag`.

The values of `ctxTable`, `ctxIdx` and `bypassFlag` are derived as follows based on the entries for `binIdx` of the corresponding syntax element in Table 9-48:

- If the entry in Table 9-48 is not equal to "bypass", "terminate" or "na", the values of `binIdx` are decoded by invoking the `DecodeDecision` process as specified in clause 9.3.4.3.2 and the following applies:
 - `ctxTable` is specified in Table 9-4.
 - The variable `ctxInc` is specified by the corresponding entry in Table 9-48 and when more than one value is listed in Table 9-48 for a `binIdx`, the assignment process for `ctxInc` for that `binIdx` is further specified in the clauses given in parenthesis.
 - The variable `ctxIdxOffset` is specified by the lowest value of `ctxIdx` in Table 9-4 depending on the current value of `initType`.
 - `ctxIdx` is set equal to the sum of `ctxInc` and `ctxIdxOffset`.
 - `bypassFlag` is set equal to 0.
- Otherwise, if the entry in Table 9-48 is equal to "bypass", the values of `binIdx` are decoded by invoking the `DecodeBypass` process as specified in clause 9.3.4.3.4 and the following applies:
 - `ctxTable` is set equal to 0.

- ctxIdx is set equal to 0.
- bypassFlag is set equal to 1.
- Otherwise, if the entry in Table 9-48 is equal to "terminate", the values of binIdx are decoded by invoking the DecodeTerminate process as specified in clause 9.3.4.3.5 and the following applies:
 - ctxTable is set equal to 0.
 - ctxIdx is set equal to 0.
 - bypassFlag is set equal to 0.
- Otherwise (the entry in Table 9-48 is equal to "na"), the values of binIdx do not occur for the corresponding syntax element.

Table 9-48 – Assignment of ctxInc to syntax elements with context coded bins

Syntax element	binIdx					
	0	1	2	3	4	>= 5
end of slice segment flag	terminate	na	na	na	na	na
end of subset one bit	terminate	na	na	na	na	na
sao merge left flag	0	na	na	na	na	na
sao merge up flag	0	na	na	na	na	na
sao type idx luma	0	bypass	na	na	na	na
sao type idx chroma	0	bypass	na	na	na	na
sao_offset_abs[][][]	bypass	bypass	bypass	bypass	bypass	na
sao_offset_sign[][][]	bypass	na	na	na	na	na
sao_band_position[][][]	bypass	bypass	bypass	bypass	bypass	bypass
sao_eo_class luma	bypass	bypass	na	na	na	na
sao_eo_class chroma	bypass	bypass	na	na	na	na
split_cu_flag[][]	0,1,2 (clause 9.3.4.2.2)	na	na	na	na	na
cu_transquant_bypass_flag	0	na	na	na	na	na
cu_skip_flag	0,1,2 (clause 9.3.4.2.2)	na	na	na	na	na
palette_mode_flag	0	na	na	na	na	na
pred_mode_flag	0	na	na	na	na	na
part_mode log2CbSize == MinCbLog2SizeY	0	1	2	bypass	na	na
part_mode log2CbSize > MinCbLog2SizeY	0	1	3	bypass	na	na
pcm_flag[][]	terminate	na	na	na	na	na
prev_intra_luma_pred_flag[][]	0	na	na	na	na	na
mpm_idx[][]	bypass	bypass	na	na	na	na
rem_intra_luma_pred_mode[][]	bypass	bypass	bypass	bypass	bypass	na
intra_chroma_pred_mode[][]	0	bypass	bypass	na	na	na
rqt_root_cbf	0	na	na	na	na	na
tu_residual_act_flag	0	na	na	na	na	na
merge_flag[][]	0	na	na	na	na	na
merge_idx[][]	0	bypass	bypass	bypass	na	na
inter_pred_idc[x0][y0]	(nPbW + nPbH) != 12 ? CtDepth[x0][y0] : 4	4	na	na	na	na

Table 9-48 – Assignment of ctxInc to syntax elements with context coded bins

Syntax element	binIdx					
	0	1	2	3	4	>= 5
ref_idx_l0[][]	0	1	bypass	bypass	bypass	bypass
ref_idx_l1[][]	0	1	bypass	bypass	bypass	bypass
mvp_l0_flag[][]	0	na	na	na	na	na
mvp_l1_flag[][]	0	na	na	na	na	na
split_transform_flag[][][]	$5 - \log_2 \text{TrafoSize}$	na	na	na	na	na
cbf_cb[][][]	trafoDepth	na	na	na	na	na
cbf_cr[][][]	trafoDepth	na	na	na	na	na
cbf_luma[][][]	trafoDepth = 0 ? 1 : 0	na	na	na	na	na
abs_mvd_greater0_flag[]	0	na	na	na	na	na
abs_mvd_greater1_flag[]	0	na	na	na	na	na
abs_mvd_minus2[]	bypass	bypass	bypass	bypass	bypass	bypass
mvd_sign_flag[]	bypass	na	na	na	na	na
cu_qp_delta_abs	0	1	1	1	1	bypass
cu_qp_delta_sign_flag	bypass	na	na	na	na	na
cu_chroma_qp_offset_flag	0	na	na	na	na	na
cu_chroma_qp_offset_idx	0	0	0	0	0	na
log2_res_scale_abs_plus1[c]	$4 * c + 0$	$4 * c + 1$	$4 * c + 2$	$4 * c + 3$	na	na
res_scale_sign_flag[c]	c	na	na	na	na	na
transform_skip_flag[][][]	0	na	na	na	na	na
explicit_rdpem_flag[][][]	0	na	na	na	na	na
explicit_rdpem_dir_flag[][][]	0	na	na	na	na	na
last_sig_coeff_x_prefix	0..17 (clause 9.3.4.2.3)					
last_sig_coeff_y_prefix	0..17 (clause 9.3.4.2.3)					
last_sig_coeff_x_suffix	bypass	bypass	bypass	na	na	na
last_sig_coeff_y_suffix	bypass	bypass	bypass	na	na	na
coded_sub_block_flag[][]	0..3 (clause 9.3.4.2.4)	na	na	na	na	na
sig_coeff_flag[][]	0..43 (clause 9.3.4.2.5)	na	na	na	na	na
coeff_abs_level_greater1_flag[]	0..23 (clause 9.3.4.2.6)	na	na	na	na	na
coeff_abs_level_greater2_flag[]	0..5 (clause 9.3.4.2.7)	na	na	na	na	na
coeff_abs_level_remaining[]	bypass	bypass	bypass	bypass	bypass	bypass
coeff_sign_flag[]	bypass	na	na	na	na	na
palette_predictor_run	bypass	bypass	bypass	bypass	bypass	bypass
num_signalled_palette_entries	bypass	bypass	bypass	bypass	bypass	bypass
new_palette_entries	bypass	bypass	bypass	bypass	bypass	bypass
palette_escape_val_present_flag	bypass	na	na	na	na	na
palette_transpose_flag	0	na	na	na	na	na
num_palette_indices_minus1	bypass	bypass	bypass	bypass	bypass	bypass

Table 9-48 – Assignment of ctxInc to syntax elements with context coded bins

Syntax element	binIdx					
	0	1	2	3	4	>= 5
palette_idx_idc	bypass	bypass	bypass	bypass	bypass	bypass
copy_above_palette_indices_flag and copy above indices for final run flag	0	na	na	na	na	na
palette_run_prefix	0..7 (clause 9.3.4.2.8)					
palette_run_suffix	bypass	bypass	bypass	bypass	bypass	bypass
palette_escape_val	bypass	bypass	bypass	bypass	bypass	bypass

9.3.4.2.2 Derivation process of ctxInc using left and above syntax elements

Input to this process is the luma location (x0, y0) specifying the top-left luma sample of the current luma block relative to the top-left sample of the current picture.

Output of this process is ctxInc.

The location (xNbL, yNbL) is set equal to (x0 – 1, y0) and the variable availableL, specifying the availability of the block located directly to the left of the current block, is derived by invoking the availability derivation process for a block in z-scan order as specified in clause 6.4.1 with the location (xCurr, yCurr) set equal to (x0, y0) and the neighbouring location (xNbY, yNbY) set equal to (xNbL, yNbL) as inputs, and the output is assigned to availableL.

The location (xNbA, yNbA) is set equal to (x0, y0 – 1) and the variable availableA specifying the availability of the coding block located directly above the current block, is derived by invoking the availability derivation process for a block in z-scan order as specified in clause 6.4.1 with the location (xCurr, yCurr) set equal to (x0, y0) and the neighbouring location (xNbY, yNbY) set equal to (xNbA, yNbA) as inputs, and the output is assigned to availableA.

The assignment of ctxInc for the syntax elements split_cu_flag[x0][y0] and cu_skip_flag[x0][y0] is specified in Table 9-49.

Table 9-49 – Specification of ctxInc using left and above syntax elements

Syntax element	condL	condA	ctxInc
split_cu_flag[x0][y0]	CtDepth[xNbL][yNbL] > cqtDepth	CtDepth[xNbA][yNbA] > cqtDepth	(condL && availableL) + (condA && availableA)
cu_skip_flag[x0][y0]	cu_skip_flag[xNbL][yNbL]	cu_skip_flag[xNbA][yNbA]	(condL && availableL) + (condA && availableA)

9.3.4.2.3 Derivation process of ctxInc for the syntax elements last_sig_coeff_x_prefix and last_sig_coeff_y_prefix

Inputs to this process are the variable binIdx, the colour component index cIdx and the transform block size log2TrafoSize.

Output of this process is the variable ctxInc.

The variables ctxOffset and ctxShift are derived as follows:

- If cIdx is equal to 0, ctxOffset is set equal to $3 * (\log_2 \text{TrafoSize} - 2) + ((\log_2 \text{TrafoSize} - 1) \gg 2)$ and ctxShift is set equal to $(\log_2 \text{TrafoSize} + 1) \gg 2$.
- Otherwise (cIdx is greater than 0), ctxOffset is set equal to 15 and ctxShift is set equal to $\log_2 \text{TrafoSize} - 2$.

The variable ctxInc is derived as follows:

$$\text{ctxInc} = (\text{binIdx} \gg \text{ctxShift}) + \text{ctxOffset} \quad (9-34)$$

9.3.4.2.4 Derivation process of ctxInc for the syntax element coded_sub_block_flag

Inputs to this process are the colour component index cIdx, the current sub-block scan location (xS, yS), the previously decoded bins of the syntax element coded_sub_block_flag and the transform block size log2TrafoSize.

Output of this process is the variable `ctxInc`.

The variable `csbfCtx` is derived using the current location (`xS`, `yS`), two previously decoded bins of the syntax element `coded_sub_block_flag` in scan order and the transform block size `log2TrafoSize`, as follows:

- `csbfCtx` is initialized with 0 as follows:

$$\text{csbfCtx} = 0 \quad (9-35)$$

- When `xS` is less than $(1 \ll (\log_2\text{TrafoSize} - 2)) - 1$, `csbfCtx` is modified as follows:

$$\text{csbfCtx} += \text{coded_sub_block_flag}[xS + 1][yS] \quad (9-36)$$

- When `yS` is less than $(1 \ll (\log_2\text{TrafoSize} - 2)) - 1$, `csbfCtx` is modified as follows:

$$\text{csbfCtx} += \text{coded_sub_block_flag}[xS][yS + 1] \quad (9-37)$$

The context index increment `ctxInc` is derived using the colour component index `cIdx` and `csbfCtx` as follows:

- If `cIdx` is equal to 0, `ctxInc` is derived as follows:

$$\text{ctxInc} = \text{Min}(\text{csbfCtx}, 1) \quad (9-38)$$

- Otherwise (`cIdx` is greater than 0), `ctxInc` is derived as follows:

$$\text{ctxInc} = 2 + \text{Min}(\text{csbfCtx}, 1) \quad (9-39)$$

9.3.4.2.5 Derivation process of `ctxInc` for the syntax element `sig_coeff_flag`

Inputs to this process are the colour component index `cIdx`, the luma location (`x0`, `y0`) specifying the top-left sample of the current transform block relative to the top-left sample of the current picture, the current coefficient scan location (`xC`, `yC`), the scan order index `scanIdx` and the transform block size `log2TrafoSize`.

Output of this process is the variable `ctxInc`.

The variable `sigCtx` depends on the current location (`xC`, `yC`), the colour component index `cIdx`, the value of `transform_skip_flag`, the value of `cu_transquant_bypass_flag`, the transform block size and previously decoded bins of the syntax element `coded_sub_block_flag`. For the derivation of `sigCtx`, the following applies:

- If `transform_skip_context_enabled_flag` is equal to 1 and either or both `transform_skip_flag[x0][y0][cIdx]` is equal to 1 or `cu_transquant_bypass_flag` is equal to 1, `sigCtx` is derived as follows:

$$\text{sigCtx} = (\text{cIdx} == 0) ? 42 : 16 \quad (9-40)$$

- Otherwise, if `log2TrafoSize` is equal to 2, `sigCtx` is derived using `ctxIdxMap[]` specified in Table 9-50 as follows:

$$\text{sigCtx} = \text{ctxIdxMap}[(yC \ll 2) + xC] \quad (9-41)$$

- Otherwise, if `xC + yC` is equal to 0, `sigCtx` is derived as follows:

$$\text{sigCtx} = 0 \quad (9-42)$$

- Otherwise, `sigCtx` is derived using previous values of `coded_sub_block_flag` as follows:

- The sub-block location (`xS`, `yS`) is set equal to (`xC >> 2`, `yC >> 2`).

- The variable `prevCsbF` is set equal to 0.

- When `xS` is less than $(1 \ll (\log_2\text{TrafoSize} - 2)) - 1$, the following applies:

$$\text{prevCsbF} += \text{coded_sub_block_flag}[xS + 1][yS] \quad (9-43)$$

- When `yS` is less than $(1 \ll (\log_2\text{TrafoSize} - 2)) - 1$, the following applies:

$$\text{prevCsbF} += (\text{coded_sub_block_flag}[xS][yS + 1] \ll 1) \quad (9-44)$$

- The inner sub-block location (`xP`, `yP`) is set equal to (`xC & 3`, `yC & 3`).

- The variable sigCtx is derived as follows:

- If prevCsbF is equal to 0, the following applies:

$$\text{sigCtx} = (\text{xP} + \text{yP} == 0) ? 2 : (\text{xP} + \text{yP} < 3) ? 1 : 0 \quad (9-45)$$

- Otherwise, if prevCsbF is equal to 1, the following applies:

$$\text{sigCtx} = (\text{yP} == 0) ? 2 : (\text{yP} == 1) ? 1 : 0 \quad (9-46)$$

- Otherwise, if prevCsbF is equal to 2, the following applies:

$$\text{sigCtx} = (\text{xP} == 0) ? 2 : (\text{xP} == 1) ? 1 : 0 \quad (9-47)$$

- Otherwise (prevCsbF is equal to 3), the following applies:

$$\text{sigCtx} = 2 \quad (9-48)$$

- The variable sigCtx is modified as follows:

- If cIdx is equal to 0, the following applies:

- When (xS + yS) is greater than 0, the following applies:

$$\text{sigCtx} += 3 \quad (9-49)$$

- The variable sigCtx is modified as follows:

- If log2TrafoSize is equal to 3, the following applies:

$$\text{sigCtx} += (\text{scanIdx} == 0) ? 9 : 15 \quad (9-50)$$

- Otherwise, the following applies:

$$\text{sigCtx} += 21 \quad (9-51)$$

- Otherwise (cIdx is greater than 0), the following applies:

- If log2TrafoSize is equal to 3, the following applies:

$$\text{sigCtx} += 9 \quad (9-52)$$

- Otherwise, the following applies:

$$\text{sigCtx} += 12 \quad (9-53)$$

The context index increment ctxInc is derived using the colour component index cIdx and sigCtx as follows:

- If cIdx is equal to 0, ctxInc is derived as follows:

$$\text{ctxInc} = \text{sigCtx} \quad (9-54)$$

- Otherwise (cIdx is greater than 0), ctxInc is derived as follows:

$$\text{ctxInc} = 27 + \text{sigCtx} \quad (9-55)$$

Table 9-50 – Specification of ctxIdxMap[i]

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
ctxIdxMap[i]	0	1	4	5	2	3	4	5	6	6	8	8	7	7	8

9.3.4.2.6 Derivation process of ctxInc for the syntax element coeff_abs_level_greater1_flag

Inputs to this process are the colour component index cIdx, the current sub-block scan index i and the current coefficient scan index n within the current sub-block.

Output of this process is the variable ctxInc.

The variable ctxSet specifies the current context set and for its derivation the following applies:

- If this process is invoked for the first time for the current sub-block scan index i, the following applies:
 - The variable ctxSet is initialized as follows:
 - If the current sub-block scan index i is equal to 0 or cIdx is greater than 0, the following applies:
$$\text{ctxSet} = 0 \quad (9-56)$$
 - Otherwise (i is greater than 0 and cIdx is equal to 0), the following applies:
$$\text{ctxSet} = 2 \quad (9-57)$$
 - The variable lastGreater1Ctx is derived as follows:
 - If the current sub-block with scan index i is the first one to be processed in this clause for the current transform block, the variable lastGreater1Ctx is set equal to 1.
 - Otherwise, the following applies:
 - The variable lastGreater1Ctx is set equal to the value of greater1Ctx that has been derived during the last invocation of the process specified in this clause for a previous sub-block.
 - When lastGreater1Ctx is greater than 0, the variable lastGreater1Flag is set equal to the value of the syntax element coeff_abs_level_greater1_flag that has been used during the last invocation of the process specified in this clause for a previous sub-block and lastGreater1Ctx is modified as follows:
 - If lastGreater1Flag is equal to 1, lastGreater1Ctx is set equal to 0.
 - Otherwise (lastGreater1Flag is equal to 0), lastGreater1Ctx is incremented by 1.
 - When lastGreater1Ctx is equal to 0, ctxSet is incremented by one as follows:
$$\text{ctxSet} = \text{ctxSet} + 1 \quad (9-58)$$
 - The variable greater1Ctx is set equal to 1.
- Otherwise (this process is not invoked for the first time for the current sub-block scan index i), the following applies:
 - The variable ctxSet is set equal to the variable ctxSet that has been derived during the last invocation of the process specified in this clause.
 - The variable greater1Ctx is set equal to the variable greater1Ctx that has been derived during the last invocation of the process specified in this clause.
 - When greater1Ctx is greater than 0, the variable lastGreater1Flag is set equal to the syntax element coeff_abs_level_greater1_flag that has been used during the last invocation of the process specified in this clause and greater1Ctx is modified as follows:
 - If lastGreater1Flag is equal to 1, greater1Ctx is set equal to 0.
 - Otherwise (lastGreater1Flag is equal to 0), greater1Ctx is incremented by 1.

The context index increment ctxInc is derived using the current context set ctxSet and the current context greater1Ctx as follows:

$$\text{ctxInc} = (\text{ctxSet} * 4) + \text{Min}(3, \text{greater1Ctx}) \quad (9-59)$$

When cIdx is greater than 0, ctxInc is modified as follows:

$$\text{ctxInc} = \text{ctxInc} + 16 \quad (9-60)$$

9.3.4.2.7 Derivation process of ctxInc for the syntax element coeff_abs_level_greater2_flag

Inputs to this process are the colour component index cIdx, the current sub-block scan index i and the current coefficient scan index n within the current sub-block.

Output of this process is the variable ctxInc.

The variable ctxSet specifies the current context set and is set equal to the value of the variable ctxSet that has been derived in clause 9.3.4.2.6 for the same subset i.

The context index increment ctxInc is set equal to the variable ctxSet as follows:

$$\text{ctxInc} = \text{ctxSet} \quad (9-61)$$

When cIdx is greater than 0, ctxInc is modified as follows:

$$\text{ctxInc} = \text{ctxInc} + 4 \quad (9-62)$$

9.3.4.2.8 Derivation process of ctxInc for the syntax element palette_run_prefix

Inputs to this process are the bin index binIdx and the syntax elements copy_above_palette_indices_flag and palette_idx_idc.

Output of this process is the variable ctxInc.

The variable ctxInc is derived as follows:

- If copy_above_palette_indices_flag is equal to 0 and binIdx is equal to 0, ctxInc is derived as follows:

$$\text{ctxInc} = (\text{palette_idx_idc} < 1) ? 0 : ((\text{palette_idx_idc} < 3) ? 1 : 2) \quad (9-63)$$

- Otherwise, ctxInc is provided by Table 9-51.

Table 9-51 – Specification of ctxIdxMap[copy_above_palette_indices_flag][binIdx]

binIdx	0	1	2	3	4	> 4
copy_above_palette_indices_flag == 1	5	6	6	7	7	bypass
copy_above_palette_indices_flag == 0	0, 1, 2	3	3	4	4	bypass

9.3.4.3 Arithmetic decoding process

9.3.4.3.1 General

Inputs to this process are ctxTable, ctxIdx and bypassFlag, as derived in clause 9.3.4.2, and the state variables ivlCurrRange and ivlOffset of the arithmetic decoding engine.

Output of this process is the value of the bin.

Figure 9-5 illustrates the whole arithmetic decoding process for a single bin. For decoding the value of a bin, the context index table ctxTable and the ctxIdx are passed to the arithmetic decoding process DecodeBin(ctxTable, ctxIdx), which is specified as follows:

- If bypassFlag is equal to 1, DecodeBypass() as specified in clause 9.3.4.3.4 is invoked.
- Otherwise, if bypassFlag is equal to 0, ctxTable is equal to 0 and ctxIdx is equal to 0, DecodeTerminate() as specified in clause 9.3.4.3.5 is invoked.
- Otherwise (bypassFlag is equal to 0 and ctxTable is not equal to 0), DecodeDecision() as specified in clause 9.3.4.3.2 is invoked.

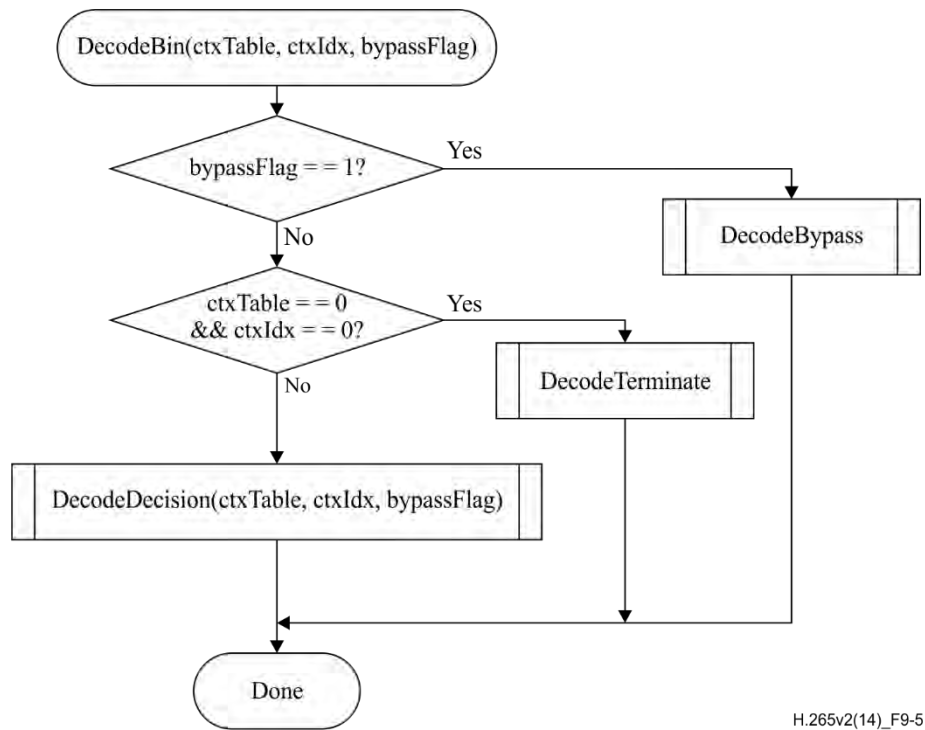


Figure 9-5 – Overview of the arithmetic decoding process for a single bin (informative)

NOTE – Arithmetic coding is based on the principle of recursive interval subdivision. Given a probability estimation $p(0)$ and $p(1) = 1 - p(0)$ of a binary decision $(0, 1)$, an initially given code sub-interval with the range $ivlCurrRange$ will be subdivided into two sub-intervals having range $p(0) * ivlCurrRange$ and $ivlCurrRange - p(0) * ivlCurrRange$, respectively. Depending on the decision, which has been observed, the corresponding sub-interval will be chosen as the new code interval, and a binary code string pointing into that interval will represent the sequence of observed binary decisions. It is useful to distinguish between the most probable symbol (MPS) and the least probable symbol (LPS), so that binary decisions have to be identified as either MPS or LPS, rather than 0 or 1. Given this terminology, each context is specified by the probability $plps$ of the LPS and the value of MPS ($valMps$), which is either 0 or 1. The arithmetic core engine in this Specification has three distinct properties:

- The probability estimation is performed by means of a finite-state machine with a table-based transition process between 64 different representative probability states $\{plps(pStateIdx) \mid 0 \leq pStateIdx < 64\}$ for the LPS probability $plps$. The numbering of the states is arranged in such a way that the probability state with index $pStateIdx = 0$ corresponds to an LPS probability value of 0.5, with decreasing LPS probability towards higher state indices.
- The range $ivlCurrRange$ representing the state of the coding engine is quantized to a small set $\{Q_1, \dots, Q_4\}$ of pre-set quantization values prior to the calculation of the new interval range. Storing a table containing all 64×4 pre-computed product values of $Q_i * plps(pStateIdx)$ allows a multiplication-free approximation of the product $ivlCurrRange * plps(pStateIdx)$.
- For syntax elements or parts thereof for which an approximately uniform probability distribution is assumed to be given a separate simplified encoding and decoding bypass process is used.

9.3.4.3.2 Arithmetic decoding process for a binary decision

9.3.4.3.2.1 General

Inputs to this process are the variables $ctxTable$, $ctxIdx$, $ivlCurrRange$ and $ivlOffset$.

Outputs of this process are the decoded value $binVal$ and the updated variables $ivlCurrRange$ and $ivlOffset$.

Figure 9-6 shows the flowchart for decoding a single decision ($DecodeDecision$):

1. The value of the variable $ivlLpsRange$ is derived as follows:

- Given the current value of $ivlCurrRange$, the variable $qRangeIdx$ is derived as follows:

$$qRangeIdx = (ivlCurrRange \gg 6) \& 3 \quad (9-64)$$

- Given $qRangeIdx$ and $pStateIdx$ associated with $ctxTable$ and $ctxIdx$, the value of the variable $rangeTabLps$ as specified in Table 9-52 is assigned to $ivlLpsRange$:

$$ivlLpsRange = rangeTabLps[pStateIdx][qRangeIdx] \quad (9-65)$$

2. The variable `ivlCurrRange` is set equal to `ivlCurrRange – ivlLpsRange` and the following applies:
 - If `ivlOffset` is greater than or equal to `ivlCurrRange`, the variable `binVal` is set equal to `1 – valMps`, `ivlOffset` is decremented by `ivlCurrRange` and `ivlCurrRange` is set equal to `ivlLpsRange`.
 - Otherwise, the variable `binVal` is set equal to `valMps`.

Given the value of `binVal`, the state transition is performed as specified in clause 9.3.4.3.2.2. Depending on the current value of `ivlCurrRange`, renormalization is performed as specified in clause 9.3.4.3.3.

9.3.4.3.2.2 State transition process

Inputs to this process are the current `pStateIdx`, the decoded value `binVal` and `valMps` values of the context variable associated with `ctxTable` and `ctxIdx`.

Outputs of this process are the updated `pStateIdx` and `valMps` of the context variable associated with `ctxIdx`.

Depending on the decoded value `binVal`, the update of the two variables `pStateIdx` and `valMps` associated with `ctxIdx` is derived as follows:

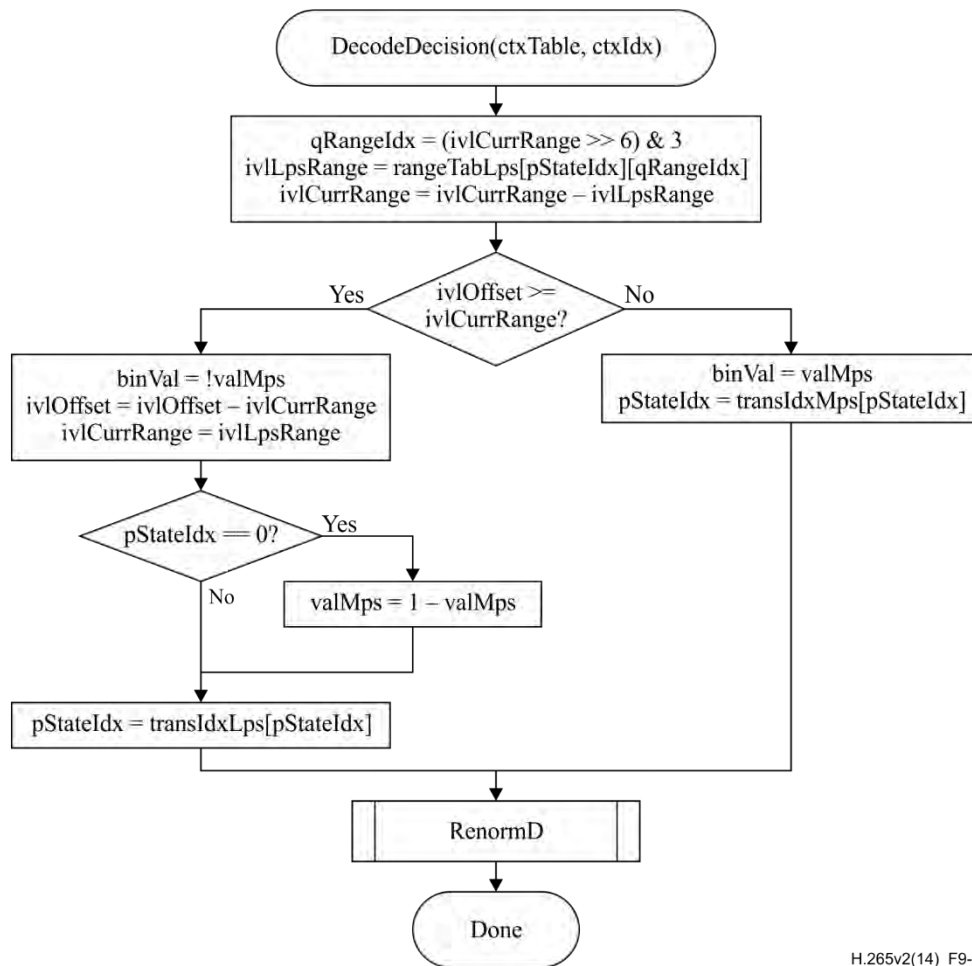
```

if( binVal == valMps )
    pStateIdx = transIdxMps( pStateIdx )
else {
    if( pStateIdx == 0 )
        valMps = 1 – valMps
    pStateIdx = transIdxLps( pStateIdx )
}

```

(9-66)

Table 9-53 specifies the transition rules `transIdxMps()` and `transIdxLps()` after decoding the value of `valMps` and `1 – valMps`, respectively.



H.265v2(14)_F9-6

Figure 9-6 – Flowchart for decoding a decision

Table 9-52 – Specification of rangeTabLps depending on the values of pStateIdx and qRangeIdx

pStateIdx	qRangeIdx				pStateIdx	qRangeIdx			
	0	1	2	3		0	1	2	3
0	128	176	208	240	32	27	33	39	45
1	128	167	197	227	33	26	31	37	43
2	128	158	187	216	34	24	30	35	41
3	123	150	178	205	35	23	28	33	39
4	116	142	169	195	36	22	27	32	37
5	111	135	160	185	37	21	26	30	35
6	105	128	152	175	38	20	24	29	33
7	100	122	144	166	39	19	23	27	31
8	95	116	137	158	40	18	22	26	30
9	90	110	130	150	41	17	21	25	28
10	85	104	123	142	42	16	20	23	27
11	81	99	117	135	43	15	19	22	25
12	77	94	111	128	44	14	18	21	24
13	73	89	105	122	45	14	17	20	23
14	69	85	100	116	46	13	16	19	22
15	66	80	95	110	47	12	15	18	21
16	62	76	90	104	48	12	14	17	20
17	59	72	86	99	49	11	14	16	19
18	56	69	81	94	50	11	13	15	18
19	53	65	77	89	51	10	12	15	17
20	51	62	73	85	52	10	12	14	16
21	48	59	69	80	53	9	11	13	15
22	46	56	66	76	54	9	11	12	14
23	43	53	63	72	55	8	10	12	14
24	41	50	59	69	56	8	9	11	13
25	39	48	56	65	57	7	9	11	12
26	37	45	54	62	58	7	9	10	12
27	35	43	51	59	59	7	8	10	11
28	33	41	48	56	60	6	8	9	11
29	32	39	46	53	61	6	7	9	10
30	30	37	43	50	62	6	7	8	9
31	29	35	41	48	63	2	2	2	2

Table 9-53 – State transition table

pStateIdx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
transIdxLps	0	0	1	2	2	4	4	5	6	7	8	9	9	11	11	12
transIdxMps	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
pStateIdx	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
transIdxLps	13	13	15	15	16	16	18	18	19	19	21	21	22	22	23	24
transIdxMps	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
pStateIdx	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
transIdxLps	24	25	26	26	27	27	28	29	29	30	30	30	31	32	32	33
transIdxMps	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
pStateIdx	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
transIdxLps	33	33	34	34	35	35	35	36	36	36	37	37	37	38	38	63
transIdxMps	49	50	51	52	53	54	55	56	57	58	59	60	61	62	62	63

9.3.4.3.3 Renormalization process in the arithmetic decoding engine

Inputs to this process are bits from slice segment data and the variables ivlCurrRange and ivlOffset.

Outputs of this process are the updated variables ivlCurrRange and ivlOffset.

A flowchart of the renormalization is shown in Figure 9-7. The current value of ivlCurrRange is first compared to 256 and then the following applies:

- If ivlCurrRange is greater than or equal to 256, no renormalization is needed and the RenormD process is finished;
- Otherwise (ivlCurrRange is less than 256), the renormalization loop is entered. Within this loop, the value of ivlCurrRange is doubled, i.e., left-shifted by 1 and a single bit is shifted into ivlOffset by using read_bits(1).

The bitstream shall not contain data that result in a value of ivlOffset being greater than or equal to ivlCurrRange upon completion of this process.

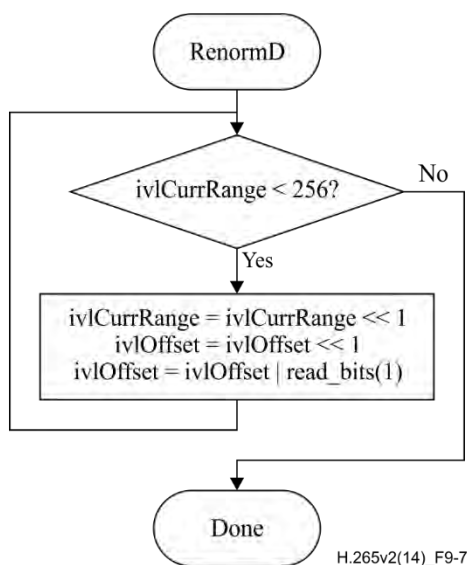


Figure 9-7 – Flowchart of renormalization

9.3.4.3.4 Bypass decoding process for binary decisions

Inputs to this process are bits from slice segment data and the variables `ivlCurrRange` and `ivlOffset`.

Outputs of this process are the updated variable `ivlOffset` and the decoded value `binVal`.

The bypass decoding process is invoked when `bypassFlag` is equal to 1. Figure 9-8 shows a flowchart of the corresponding process.

First, the value of `ivlOffset` is doubled, i.e., left-shifted by 1 and a single bit is shifted into `ivlOffset` by using `read_bits(1)`. Then, the value of `ivlOffset` is compared to the value of `ivlCurrRange` and then the following applies:

- If `ivlOffset` is greater than or equal to `ivlCurrRange`, the variable `binVal` is set equal to 1 and `ivlOffset` is decremented by `ivlCurrRange`.
- Otherwise (`ivlOffset` is less than `ivlCurrRange`), the variable `binVal` is set equal to 0.

The bitstream shall not contain data that result in a value of `ivlOffset` being greater than or equal to `ivlCurrRange` upon completion of this process.

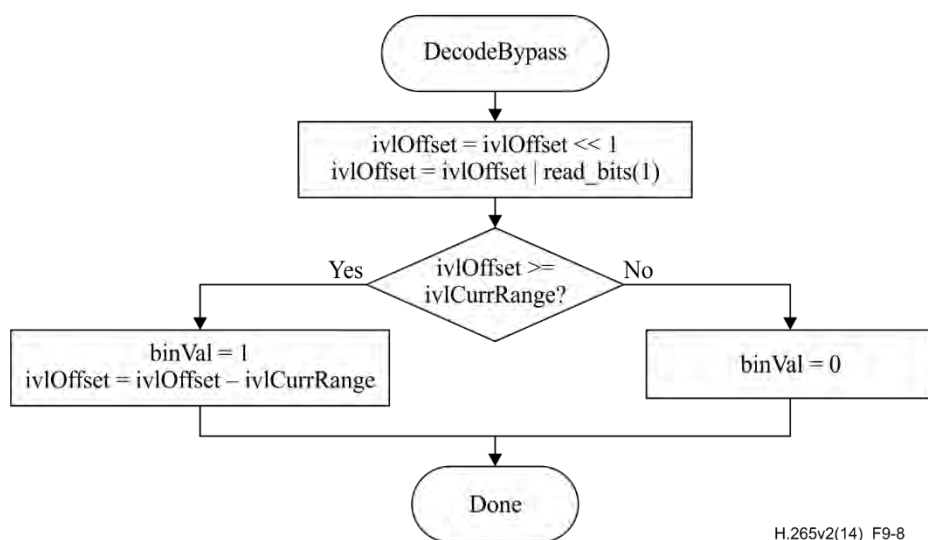


Figure 9-8 – Flowchart of bypass decoding process

9.3.4.3.5 Decoding process for binary decisions before termination

Inputs to this process are bits from slice segment data and the variables `ivlCurrRange` and `ivlOffset`.

Outputs of this process are the updated variables `ivlCurrRange` and `ivlOffset`, and the decoded value `binVal`.

This decoding process applies to decoding of `end_of_slice_segment_flag`, `end_of_subset_one_bit` and `pcm_flag` corresponding to `ctxTable` equal to 0 and `ctxIdx` equal to 0. Figure 9-9 shows the flowchart of the corresponding decoding process, which is specified as follows:

First, the value of `ivlCurrRange` is decremented by 2. Then, the value of `ivlOffset` is compared to the value of `ivlCurrRange` and then the following applies:

- If `ivlOffset` is greater than or equal to `ivlCurrRange`, the variable `binVal` is set equal to 1, no renormalization is carried out, and CABAC decoding is terminated. The last bit inserted in register `ivlOffset` is equal to 1. When decoding `end_of_slice_segment_flag`, this last bit inserted in register `ivlOffset` is interpreted as `rsbp_stop_one_bit`. When decoding `end_of_subset_one_bit`, this last bit inserted in register `ivlOffset` is interpreted as `alignment_bit_equal_to_one`.
- Otherwise (`ivlOffset` is less than `ivlCurrRange`), the variable `binVal` is set equal to 0 and renormalization is performed as specified in clause 9.3.4.3.3.

NOTE – This procedure may also be implemented using `DecodeDecision(ctxTable, ctxIdx, bypassFlag)` with `ctxTable = 0`, `ctxIdx = 0` and `bypassFlag = 0`. In the case where the decoded value is equal to 1, seven more bits would be read by `DecodeDecision(ctxTable, ctxIdx, bypassFlag)` and a decoding process would have to adjust its bitstream pointer accordingly to properly decode following syntax elements.

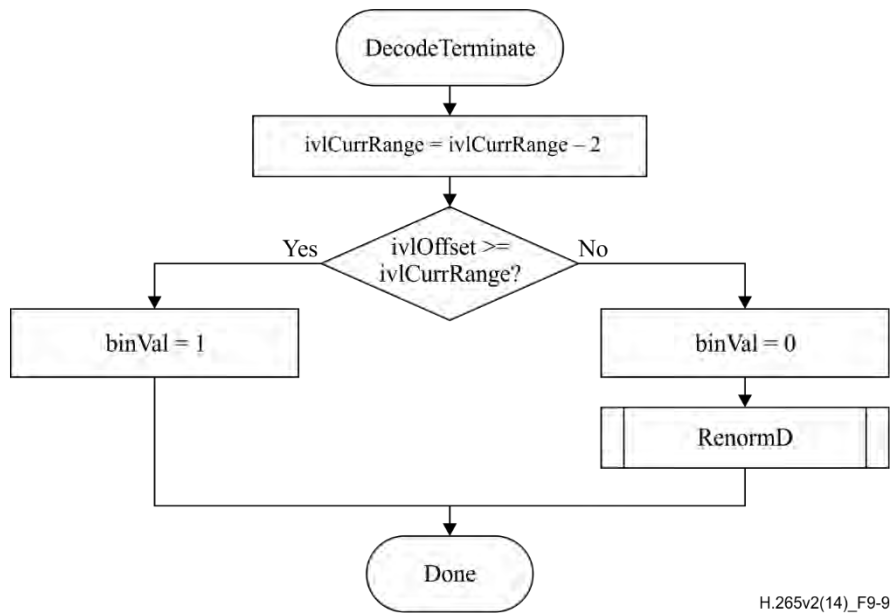


Figure 9-9 – Flowchart of decoding a decision before termination

9.3.4.3.6 Alignment process prior to aligned bypass decoding

Input to this process is the variable `ivlCurrRange`.

Output of this process is the updated variable `ivlCurrRange`.

This process applies prior to the decoding of syntax elements `coeff_abs_level_remaining[]` and `coeff_sign_flag[]`.

`ivlCurrRange` is set equal to 256.

NOTE – When `ivlCurrRange` is 256, `ivlOffset` and the bit-stream can be considered as a shift register, and `binVal` as the register's second most significant bit (the most significant bit is always 0 due to the restriction of `ivlOffset` being less than `ivlCurrRange`).

9.3.5 Arithmetic encoding process (informative)

9.3.5.1 General

This clause does not form an integral part of this Specification.

Inputs to this process are decisions that are to be encoded and written.

Outputs of this process are bits that are written to the RBSP.

This informative clause describes an arithmetic encoding engine that matches the arithmetic decoding engine described in clause 9.3.4.3. The encoding engine is essentially symmetric with the decoding engine, i.e., procedures are called in the same order. The following procedures are described in this clause: `InitEncoder`, `EncodeDecision`, `EncodeBypass`, `EncodeTerminate`, which correspond to `InitDecoder`, `DecodeDecision`, `DecodeBypass` and `DecodeTerminate`, respectively. The state of the arithmetic encoding engine is represented by a value of the variable `ivlLow` pointing to the lower end of a sub-interval and a value of the variable `ivlCurrRange` specifying the corresponding range of that sub-interval.

9.3.5.2 Initialization process for the arithmetic encoding engine (informative)

This clause does not form an integral part of this Specification.

This process is invoked before encoding the first coding block of a slice segment, and after encoding any `pcm_alignment_zero_bit` and all `pcm_sample_luma` and `pcm_sample_chroma` data for a coding unit with `pcm_flag` equal to 1.

Outputs of this process are the values `ivlLow`, `ivlCurrRange`, `firstBitFlag`, `bitsOutstanding` and `BinCountsInNalUnits` of the arithmetic encoding engine.

In the initialization procedure of the encoder, `ivlLow` is set equal to 0 and `ivlCurrRange` is set equal to 510. Furthermore, `firstBitFlag` is set equal to 1 and the counter `bitsOutstanding` is set equal to 0.

Depending on whether the current slice segment is the first slice segment of a coded picture, the following applies:

- If the current slice segment is the first slice segment of a coded picture, the counter `BinCountsInNalUnits` is set equal to 0.
- Otherwise (the current slice segment is not the first slice segment of a coded picture), the counter `BinCountsInNalUnits` is not modified. The value of `BinCountsInNalUnits` is the result of encoding all the slice segments of a coded picture that precede the current slice segment in decoding order. After initializing for the first slice segment of a coded picture as specified in this clause, `BinCountsInNalUnits` is incremented as specified in clauses 9.3.5.3, 9.3.5.5 and 9.3.5.6.

NOTE – The minimum register precision required for storing the values of the variables `ivlLow` and `ivlCurrRange` after invocation of any of the arithmetic encoding processes specified in clauses 9.3.5.3, 9.3.5.5 and 9.3.5.6 is 10 bits and 9 bits, respectively. The encoding process for a binary decision (`EncodeDecision`) as specified in clause 9.3.5.3 and the encoding process for a binary decision before termination (`EncodeTerminate`) as specified in clause 9.3.5.6 require a minimum register precision of 10 bits for the variable `ivlLow` and a minimum register precision of 9 bits for the variable `ivlCurrRange`. The bypass encoding process for binary decisions (`EncodeBypass`) as specified in clause 9.3.5.5 requires a minimum register precision of 11 bits for the variable `ivlLow` and a minimum register precision of 9 bits for the variable `ivlCurrRange`. The precision required for the counters `bitsOutstanding` and `BinCountsInNalUnits` should be sufficiently large to prevent overflow of the related registers. When `maxBinCountInSlice` denotes the maximum total number of binary decisions to encode in one slice segment and `maxBinCountInPic` denotes the maximum total number of binary decisions to encode a picture, the minimum register precision required for the variables `bitsOutstanding` and `BinCountsInNalUnits` is given by $\text{Ceil}(\text{Log2}(\text{maxBinCountInSlice} + 1))$ and $\text{Ceil}(\text{Log2}(\text{maxBinCountInPic} + 1))$, respectively.

9.3.5.3 Encoding process for a binary decision (informative)

This clause does not form an integral part of this Specification.

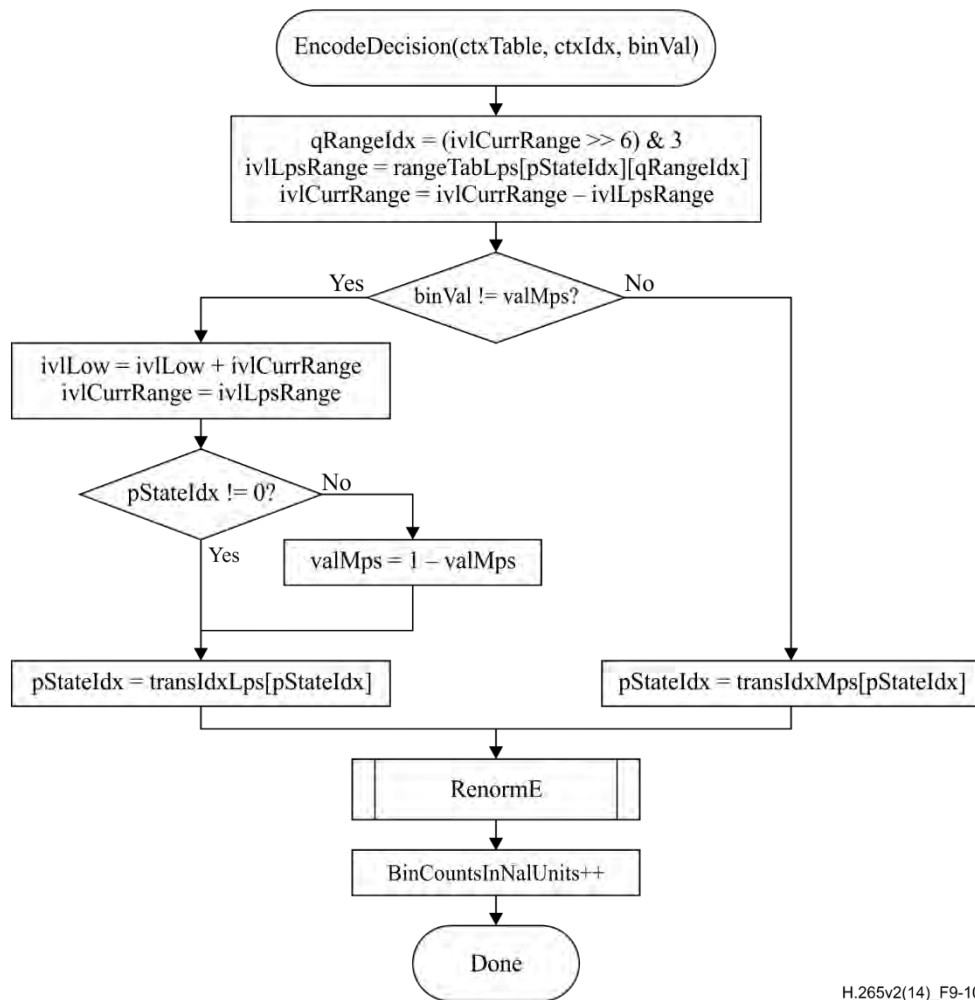
Inputs to this process are the context index `ctxIdx`, the value of `binVal` to be encoded and the variables `ivlCurrRange`, `ivlLow` and `BinCountsInNalUnits`.

Outputs of this process are the variables `ivlCurrRange`, `ivlLow` and `BinCountsInNalUnits`.

Figure 9-10 shows the flowchart for encoding a single decision. In a first step, the variable `ivlLpsRange` is derived as follows:

Given the current value of `ivlCurrRange`, `ivlCurrRange` is mapped to the index `qRangeIdx` of a quantized value of `ivlCurrRange` by using Equation 9-64. The value of `qRangeIdx` and the value of `pStateIdx` associated with `ctxIdx` are used to determine the value of the variable `rangeTabLps` as specified in Table 9-52, which is assigned to `ivlLpsRange`. The value of `ivlCurrRange` – `ivlLpsRange` is assigned to `ivlCurrRange`.

In a second step, the value of `binVal` is compared to `valMps` associated with `ctxIdx`. When `binVal` is different from `valMps`, `ivlCurrRange` is added to `ivlLow` and `ivlCurrRange` is set equal to the value `ivlLpsRange`. Given the encoded decision, the state transition is performed as specified in clause 9.3.4.3.2.2. Depending on the current value of `ivlCurrRange`, renormalization is performed as specified in clause 9.3.5.4. Finally, the variable `BinCountsInNalUnits` is incremented by 1.



H.265v2(14)_F9-10

Figure 9-10 – Flowchart for encoding a decision

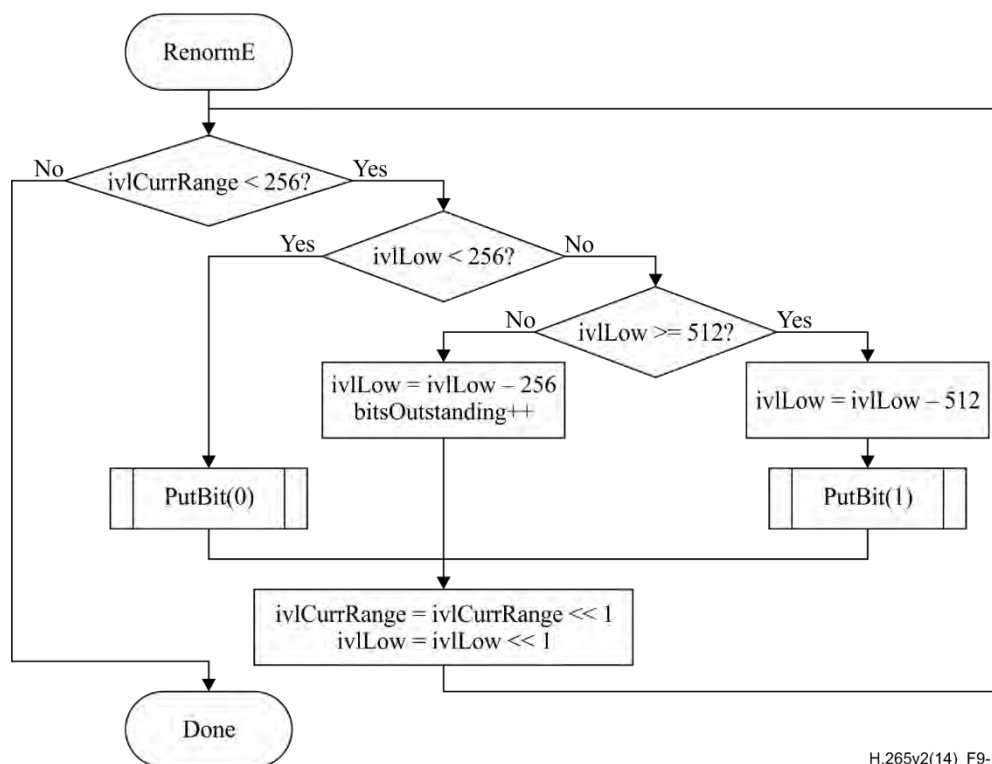
9.3.5.4 Renormalization process in the arithmetic encoding engine (informative)

This clause does not form an integral part of this Specification.

Inputs to this process are the variables ivlCurrRange, ivlLow, firstBitFlag and bitsOutstanding.

Outputs of this process are zero or more bits written to the RBSP and the updated variables ivlCurrRange, ivlLow, firstBitFlag and bitsOutstanding.

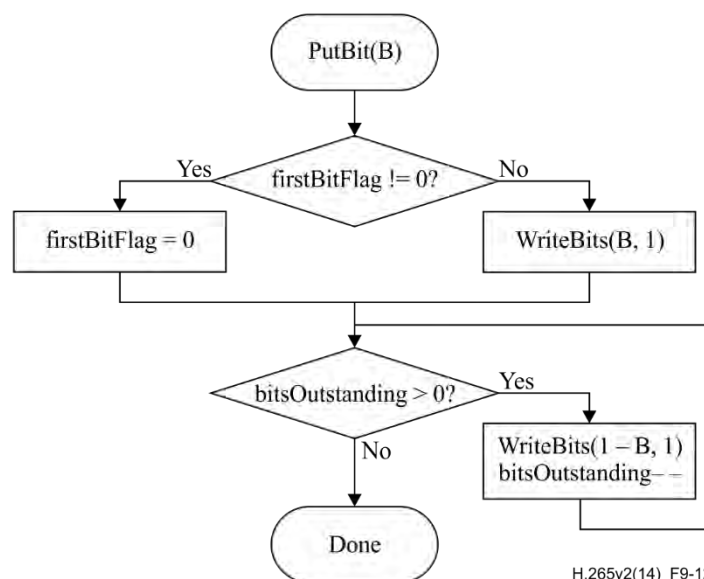
Renormalization in the encoder is illustrated in Figure 9-11.



H.265v2(14)_F9-11

Figure 9-11 – Flowchart of renormalization in the encoder

The PutBit() procedure described in Figure 9-12 provides carry over control. It uses the function WriteBits(B, N) that writes N bits with value B to the bitstream and advances the bitstream pointer by N bit positions. This function assumes the existence of a bitstream pointer with an indication of the position of the next bit to be written to the bitstream by the encoding process.



H.265v2(14)_F9-12

Figure 9-12 – Flowchart of PutBit(B)

9.3.5.5 Bypass encoding process for binary decisions (informative)

This clause does not form an integral part of this Specification.

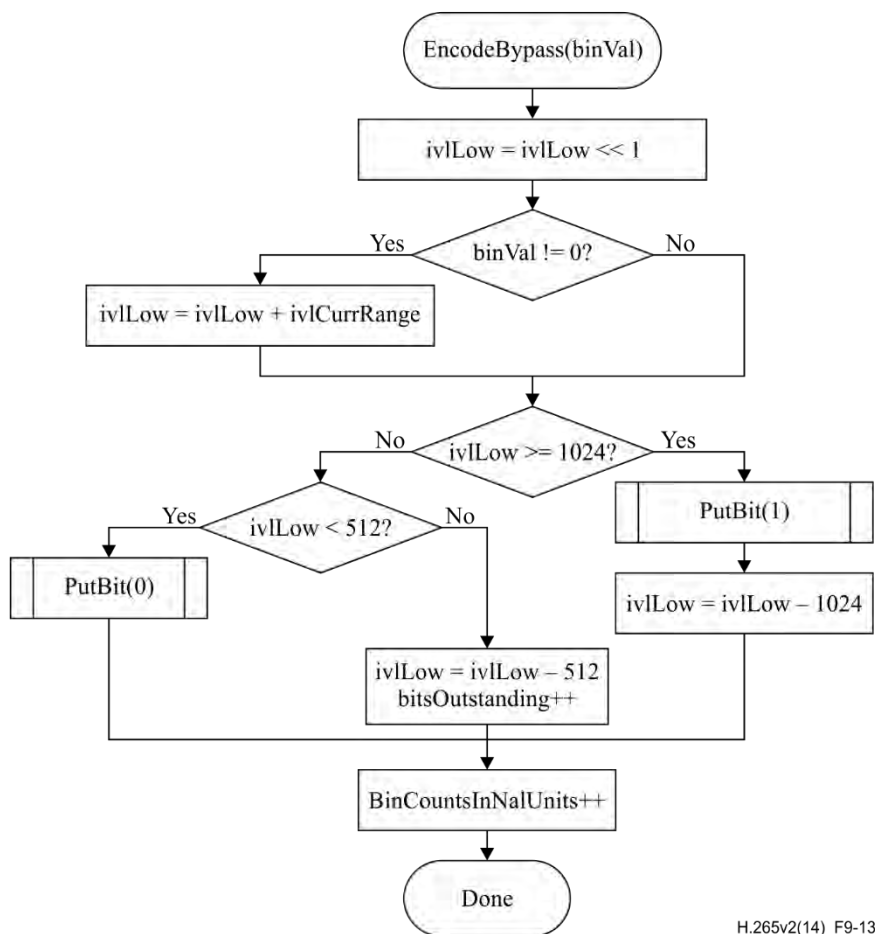
Inputs to this process are the variables binVal, ivlLow, ivlCurrRange, bitsOutstanding and BinCountsInNalUnits.

Output of this process is a bit written to the RBSP and the updated variables `ivlLow`, `bitsOutstanding` and `BinCountsInNalUnits`.

This encoding process applies to all binary decisions with `bypassFlag` equal to 1.

When `cabac_bypass_alignment_enabled_flag` is equal to 1 and `coeff_abs_level_remaining[]` is present for any coefficients in the current sub-block, an alignment process is performed. This alignment process applies prior to the encoding of the syntax elements `coeff_abs_level_remaining[]` and `coeff_sign_flag[]` and sets `ivlCurrRange` to 256.

Renormalization is included in the specification of this bypass encoding process as given in Figure 9-13.



H.265v2(14)_F9-13

Figure 9-13 – Flowchart of encoding bypass

9.3.5.6 Encoding process for a binary decision before termination (informative)

This clause does not form an integral part of this Specification.

Inputs to this process are the variables `binVal`, `ivlCurrRange`, `ivlLow`, `bitsOutstanding` and `BinCountsInNalUnits`.

Outputs of this process are zero or more bits written to the RBSP and the updated variables `ivlLow`, `ivlCurrRange`, `bitsOutstanding` and `BinCountsInNalUnits`.

This encoding routine shown in Figure 9-14 applies to encoding of `end_of_slice_segment_flag`, `end_of_subset_one_bit`, and `pcm_flag`, all associated with `ctxIdx` equal to 0.

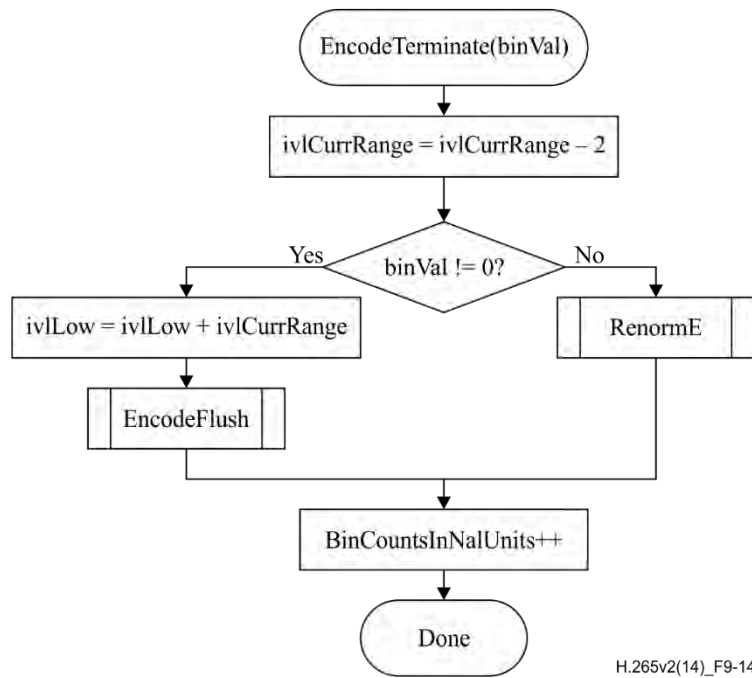


Figure 9-14 – Flowchart of encoding a decision before termination

When the value of binVal to encode is equal to 1, CABAC encoding is terminated and the flushing procedure shown in Figure 9-15 is applied. In this flushing procedure, the last bit written by WriteBits(B, N) is equal to 1. When encoding end_of_slice_segment_flag, this last bit is interpreted as rbsp_stop_one_bit. When encoding end_of_subset_one_bit, this last bit is interpreted as alignment_bit_equal_to_one.

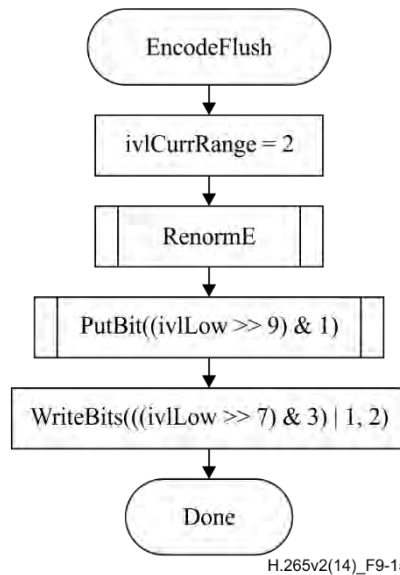


Figure 9-15 – Flowchart of flushing at termination

9.3.5.7 Byte stuffing process (informative)

This clause does not form an integral part of this Specification.

This process is invoked after encoding the last coding block of the last slice segment of a picture and after encapsulation.

Inputs to this process are the number of bytes `NumBytesInVclNalUnits` of all VCL NAL units of a picture, the number of minimum CUs `PicSizeInMinCbsY` in the picture and the number of binary symbols `BinCountsInNalUnits` resulting from encoding the contents of all VCL NAL units of the picture.

NOTE – The value of `BinCountsInNalUnits` is the result of encoding all slice segments of a coded picture. After initializing for the first slice segment of a coded picture as specified in clause 9.3.5.2, `BinCountsInNalUnits` is incremented as specified in clauses 9.3.5.3, 9.3.5.5 and 9.3.5.6.

Outputs of this process are zero or more bytes appended to the NAL unit.

Let the variable `k` be set equal to $\text{Ceil}((\text{Ceil}(3 * (32 * \text{BinCountsInNalUnits} - \text{RawMinCuBits} * \text{PicSizeInMinCbsY}) \div 1024) - \text{NumBytesInVclNalUnits}) \div 3)$. Depending on the value of `k` the following applies:

- If `k` is less than or equal to 0, no `cabac_zero_word` is appended to the NAL unit.
- Otherwise (`k` is greater than 0), the 3-byte sequence `0x000003` is appended `k` times to the NAL unit after encapsulation, where the first two bytes `0x0000` represent a `cabac_zero_word` and the third byte `0x03` represents an `emulation_prevention_three_byte`.

10 Sub-bitstream extraction process

Inputs to this process are a bitstream, a target highest `TemporalId` value `tIdTarget` and a target layer identifier list `layerIdListTarget`.

Output of this process is a sub-bitstream.

It is a requirement of bitstream conformance for the input bitstream that any output sub-bitstream that is the output of the process specified in this clause with the bitstream, `tIdTarget` equal to any value in the range of 0 to 6, inclusive, and `layerIdListTarget` either equal to the layer identifier list associated with a layer set specified in the active VPS or consisting of all the `nuh_layer_id` values of the VCL NAL units present in the input bitstream as inputs, and that satisfies both of the following conditions shall be a conforming bitstream:

- The output sub-bitstream contains at least one VCL NAL unit with `nuh_layer_id` equal to each of the `nuh_layer_id` values in `layerIdListTarget`.
- The output sub-bitstream contains at least one VCL NAL unit with `TemporalId` equal to `tIdTarget`.

NOTE 1 – A bitstream conforming to a profile specified in Annex A contains one or more coded slice segment NAL units with `nuh_layer_id` equal to 0.

NOTE 2 – A conforming bitstream contains one or more coded slice segment NAL units with `TemporalId` equal to 0.

The output sub-bitstream is derived as follows:

- When one or more of the following two conditions are true, remove all SEI NAL units that have `nuh_layer_id` equal to 0 and that contain a non-scalable-nested buffering period SEI message, a non-scalable-nested picture timing SEI message, or a non-scalable-nested decoding unit information SEI message:
 - `layerIdListTarget` does not include all the values of `nuh_layer_id` in all NAL units in the bitstream.
 - `tIdTarget` is less than the greatest `TemporalId` in all NAL units in the bitstream.
- NOTE 3 – A "smart" bitstream extractor may include appropriate non-scalable-nested buffering picture SEI messages, non-scalable-nested picture timing SEI messages and non-scalable-nested decoding unit information SEI messages in the extracted sub-bitstream, provided that the SEI messages applicable to the sub-bitstream were present as scalable-nested SEI messages in the original bitstream.
- Remove all NAL units with `TemporalId` greater than `tIdTarget` or `nuh_layer_id` not among the values included in `layerIdListTarget`.

Annex A

Profiles, tiers and levels

(This annex forms an integral part of this Recommendation | International Standard.)

A.1 Overview of profiles, tiers and levels

Profiles, tiers and levels specify restrictions on the bitstreams and hence limits on the capabilities needed to decode the bitstreams. Profiles, tiers and levels may also be used to indicate interoperability points between individual decoder implementations.

NOTE 1 – This Specification does not include individually selectable "options" at the decoder, as this would increase interoperability difficulties.

Each profile specifies a subset of algorithmic features and limits that shall be supported by all decoders conforming to that profile.

NOTE 2 – Encoders are not required to make use of any particular subset of features supported in a profile.

Each level of a tier specifies a set of limits on the values that may be taken by the syntax elements of this Specification. The same set of tier and level definitions is used with all profiles, but individual implementations may support a different tier and within a tier a different level for each supported profile. For any given profile, a level of a tier generally corresponds to a particular decoder processing load and memory capability.

The profiles that are specified in clause A.3 are also referred to as the profiles specified in Annex A.

A.2 Requirements on video decoder capability

Capabilities of video decoders conforming to this Specification are specified in terms of the ability to decode video streams conforming to the constraints of profiles, tiers and levels specified in this annex and other annexes. When expressing the capabilities of a decoder for a specified profile, the tier and level supported for that profile should also be expressed.

Specific values are specified in this annex and other annexes for the syntax elements `general_profile_idc`, `general_tier_flag`, `general_level_idc`, `sub_layer_profile_idc[i]`, `sub_layer_tier_flag[i]` and `sub_layer_level_idc[i]`. All other values of `general_profile_idc`, `general_level_idc`, `sub_layer_profile_idc[i]` and `sub_layer_level_idc[i]` are reserved for future use by ITU-T | ISO/IEC.

NOTE – Decoders should not infer that a reserved value of `general_profile_idc` or `sub_layer_profile_idc[i]` between the values specified in this Specification indicates intermediate capabilities between the specified profiles, as there are no restrictions on the method to be chosen by ITU-T | ISO/IEC for the use of such future reserved values. However, decoders should infer that a reserved value of `general_level_idc` or `sub_layer_level_idc[i]` associated with a particular value of `general_tier_flag` or `sub_layer_tier_flag[i]`, respectively, between the values specified in this Specification indicates intermediate capabilities between the specified levels of the tier.

A.3 Profiles

A.3.1 General

All constraints for PPSs that are specified are constraints for PPSs that are activated when the bitstream is decoded. All constraints for SPSs that are specified are constraints for SPSs that are activated when the bitstream is decoded.

The variable `RawCtuBits` is derived as follows:

$$\text{RawCtuBits} = \text{CtbSizeY} * \text{CtbSizeY} * \text{BitDepth}_Y + 2 * (\text{CtbWidthC} * \text{CtbHeightC}) * \text{BitDepth}_C \quad (\text{A-1})$$

A.3.2 Main profile

Bitstreams conforming to the Main profile shall obey the following constraints:

- Active VPSs shall have `vps_base_layer_internal_flag` and `vps_base_layer_available_flag` both equal to 1 only.
- Active SPSs for the base layer shall have `chroma_format_idc` equal to 1 only.
- Active SPSs for the base layer shall have `bit_depth_luma_minus8` equal to 0 only.
- Active SPSs for the base layer shall have `bit_depth_chroma_minus8` equal to 0 only.
- Active SPSs for the base layer shall have `transform_skip_rotation_enabled_flag`, `transform_skip_context_enabled_flag`, `implicit_rdpem_enabled_flag`, `explicit_rdpem_enabled_flag`,

extended_precision_processing_flag, intra_smoothing_disabled_flag, high_precision_offsets_enabled_flag, persistent_rice_adaptation_enabled_flag, cabac_bypass_alignment_enabled_flag, sps_curr_pic_ref_enabled_flag, palette_mode_enabled_flag, motion_vector_resolution_control_idc, and intra_boundary_filtering_disabled_flag, when present, equal to 0 only.

- CtbLog2SizeY derived according to active SPSs for the base layer shall be in the range of 4 to 6, inclusive.
- Active PPSs for the base layer shall have log2_max_transform_skip_block_size_minus2, chroma_qp_offset_list_enabled_flag, and residual_adaptive_colour_transform_enabled_flag, when present, equal to 0 only.
- When an active PPS for the base layer has tiles_enabled_flag equal to 1, it shall have entropy_coding_sync_enabled_flag equal to 0.
- When an active PPS for the base layer has tiles_enabled_flag equal to 1, ColumnWidthInLumaSamples[i] shall be greater than or equal to 256 for all values of i in the range of 0 to num_tile_columns_minus1, inclusive, and RowHeightInLumaSamples[j] shall be greater than or equal to 64 for all values of j in the range of 0 to num_tile_rows_minus1, inclusive.
- The number of times read_bits(1) is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing coding_tree_unit() data for any CTU shall be less than or equal to 5 * RawCtuBits / 3.
- general_level_idc and sub_layer_level_idc[i] for all values of i in active SPSs for the base layer shall not be equal to 255 (which indicates level 8.5).
- The tier and level constraints specified for the Main profile in clause A.4 shall be fulfilled.

Conformance of a bitstream to the Main profile is indicated by general_profile_idc being equal to 1 or general_profile_compatibility_flag[1] being equal to 1. Conformance of a sub-layer representation with TemporalId equal to i to the Main profile is indicated by sub_layer_profile_idc[i] being equal to 1 or sub_layer_profile_compatibility_flag[i][1] being equal to 1.

NOTE – When general_profile_compatibility_flag[1] is equal to 1, general_profile_compatibility_flag[2] should also be equal to 1. When sub_layer_profile_compatibility_flag[i][1] is equal to 1 for a value of i, sub_layer_profile_compatibility_flag[i][2] should also be equal to 1.

Decoders conforming to the Main profile at a specific level (identified by a specific value of general_level_idc) of a specific tier (identified by a specific value of general_tier_flag) shall be capable of decoding all bitstreams and sub-layer representations for which all of the following conditions apply:

- The bitstream or sub-layer representation is indicated to conform to the Main profile or the Main Still Picture profile.
- The bitstream or sub-layer representation is indicated to conform to a level that is not level 8.5 and is lower than or equal to the specified level.
- The bitstream or sub-layer representation is indicated to conform to a tier that is lower than or equal to the specified tier.

A.3.3 Main 10 and Main 10 Still Picture profiles

Bitstreams conforming to the Main 10 or Main 10 Still Picture profile shall obey the following constraints:

- In bitstreams conforming to the Main 10 Still Picture profile, the bitstream shall contain only one picture with nuh_layer_id equal to 0.
- Active VPSs shall have vps_base_layer_internal_flag and vps_base_layer_available_flag both equal to 1 only.
- Active SPSs for the base layer shall have chroma_format_idc equal to 1 only.
- Active SPSs for the base layer shall have bit_depth_luma_minus8 in the range of 0 to 2, inclusive.
- Active SPSs for the base layer shall have bit_depth_chroma_minus8 in the range of 0 to 2, inclusive.
- In bitstreams conforming to the Main 10 Still Picture profile, active SPSs for the base layer shall have sps_max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1] equal to 0 only.
- Active SPSs for the base layer shall have transform_skip_rotation_enabled_flag, transform_skip_context_enabled_flag, implicit_rdpem_enabled_flag, explicit_rdpem_enabled_flag, extended_precision_processing_flag, intra_smoothing_disabled_flag, high_precision_offsets_enabled_flag, persistent_rice_adaptation_enabled_flag, cabac_bypass_alignment_enabled_flag, sps_curr_pic_ref_enabled_flag, palette_mode_enabled_flag, motion_vector_resolution_control_idc, and intra_boundary_filtering_disabled_flag, when present, equal to 0 only.

- CtbLog2SizeY derived according to active SPSs for the base layer shall be in the range of 4 to 6, inclusive.
- Active PPSs for the base layer shall have log2_max_transform_skip_block_size_minus2, chroma_qp_offset_list_enabled_flag, and residual_adaptive_colour_transform_enabled_flag, when present, equal to 0 only.
- When an active PPS for the base layer has tiles_enabled_flag equal to 1, it shall have entropy_coding_sync_enabled_flag equal to 0.
- When an active PPS for the base layer has tiles_enabled_flag equal to 1, ColumnWidthInLumaSamples[i] shall be greater than or equal to 256 for all values of i in the range of 0 to num_tile_columns_minus1, inclusive, and RowHeightInLumaSamples[j] shall be greater than or equal to 64 for all values of j in the range of 0 to num_tile_rows_minus1, inclusive.
- The number of times read_bits(1) is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing coding_tree_unit() data for any CTU shall be less than or equal to 5 * RawCtuBits / 3.
- In bitstreams conforming to the Main 10 profile that do not conform to the Main 10 Still Picture profile, general_level_idc and sub_layer_level_idc[i] for all values of i in active SPSs for the base layer shall not be equal to 255 (which indicates level 8.5).
- The tier and level constraints specified for the Main 10 or Main 10 Still Picture profile in clause A.4, as applicable, shall be fulfilled.

Conformance of a bitstream to the Main 10 profile is indicated by general_profile_idc being equal to 2 or general_profile_compatibility_flag[2] being equal to 1. Conformance of a sub-layer representation with TemporalId equal to i to the Main 10 profile is indicated by sub_layer_profile_idc[i] being equal to 2 or sub_layer_profile_compatibility_flag[i][2] being equal to 1.

Conformance of a bitstream to the Main 10 Still Picture profile is indicated by general_one_picture_only_constraint_flag being equal to 1 together with general_profile_idc being equal to 2 or general_profile_compatibility_flag[2] being equal to 1. Conformance of a sub-layer representation with TemporalId equal to i to the Main 10 Still Picture profile is indicated by sub_layer_one_picture_only_constraint_flag being equal to 1 together with sub_layer_profile_idc[i] being equal to 2 or sub_layer_profile_compatibility_flag[i][2] being equal to 1.

NOTE – When the conformance of a bitstream to the Main 10 Still Picture profile is indicated as specified above, and the indicated level is not level 8.5, the conditions for indication of the conformance of the bitstream to the Main 10 profile are also fulfilled.

Decoders conforming to the Main 10 profile at a specific level (identified by a specific value of general_level_idc) of a specific tier (identified by a specific value of general_tier_flag) shall be capable of decoding all bitstreams and sub-layer representations for which all of the following conditions apply:

- The bitstream or sub-layer representation is indicated to conform to the Main 10 profile, the Main profile or the Main Still Picture profile.
- The bitstream or sub-layer representation is indicated to conform to a level that is not level 8.5 and is lower than or equal to the specified level.
- The bitstream or sub-layer representation is indicated to conform to a tier that is lower than or equal to the specified tier.

Decoders conforming to the Main 10 Still Picture profile at a specific level (identified by a specific value of general_level_idc) of a specific tier (identified by a specific value of general_tier_flag) shall be capable of decoding all bitstreams and sub-layer representations for which all of the following conditions apply:

- The bitstream or sub-layer representation is indicated to conform to the Main 10 Still Picture profile or the Main Still Picture profile.
- The bitstream or sub-layer representation is indicated to conform to a level that is not level 8.5 and is lower than or equal to the specified level.
- The bitstream or sub-layer representation is indicated to conform to a tier that is lower than or equal to the specified tier.

A.3.4 Main Still Picture profile

Bitstreams conforming to the Main Still Picture profile shall obey the following constraints:

- The bitstream shall contain only one picture with nuh_layer_id equal to 0.
- Active VPSs shall have vps_base_layer_internal_flag and vps_base_layer_available_flag both equal to 1 only.

- Active SPSs for the base layer shall have `chroma_format_idc` equal to 1 only.
- Active SPSs for the base layer shall have `bit_depth_luma_minus8` equal to 0 only.
- Active SPSs for the base layer shall have `bit_depth_chroma_minus8` equal to 0 only.
- Active SPSs for the base layer shall have `sps_max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1]` equal to 0 only.
- Active SPSs for the base layer shall have `transform_skip_rotation_enabled_flag`, `transform_skip_context_enabled_flag`, `implicit_rdpem_enabled_flag`, `explicit_rdpem_enabled_flag`, `extended_precision_processing_flag`, `intra_smoothing_disabled_flag`, `high_precision_offsets_enabled_flag`, `persistent_rice_adaptation_enabled_flag`, `cabac_bypass_alignment_enabled_flag`, `sps_curr_pic_ref_enabled_flag`, `palette_mode_enabled_flag`, `motion_vector_resolution_control_idc`, and `intra_boundary_filtering_disabled_flag`, when present, equal to 0 only.
- `CtbLog2SizeY` derived according to active SPSs for the base layer shall be in the range of 4 to 6, inclusive.
- Active PPSs for the base layer shall have `log2_max_transform_skip_block_size_minus2`, `chroma_qp_offset_list_enabled_flag`, and `residual_adaptive_colour_transform_enabled_flag`, when present, equal to 0 only.
- When an active PPS for the base layer has `tiles_enabled_flag` equal to 1, it shall have `entropy_coding_sync_enabled_flag` equal to 0.
- When an active PPS for the base layer has `tiles_enabled_flag` equal to 1, `ColumnWidthInLumaSamples[i]` shall be greater than or equal to 256 for all values of `i` in the range of 0 to `num_tile_columns_minus1`, inclusive, and `RowHeightInLumaSamples[j]` shall be greater than or equal to 64 for all values of `j` in the range of 0 to `num_tile_rows_minus1`, inclusive.
- The number of times `read_bits(1)` is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing `coding_tree_unit()` data for any CTU shall be less than or equal to $5 * \text{RawCtuBits} / 3$.
- The tier and level constraints specified for the Main Still Picture profile in clause A.4 shall be fulfilled.

Conformance of a bitstream to the Main Still Picture profile is indicated by `general_profile_idc` being equal to 3 or `general_profile_compatibility_flag[3]` being equal to 1.

NOTE – When `general_profile_compatibility_flag[3]` is equal to 1, `general_profile_compatibility_flag[1]` and `general_profile_compatibility_flag[2]` should also be equal to 1. When `sub_layer_profile_compatibility_flag[i][3]` is equal to 1 for a value of `i`, `sub_layer_profile_compatibility_flag[i][1]` and `sub_layer_profile_compatibility_flag[i][2]` should also be equal to 1.

Decoders conforming to the Main Still Picture profile at a specific level (identified by a specific value of `general_level_idc`) of a specific tier (identified by a specific value of `general_tier_flag`) shall be capable of decoding all bitstreams for which all of the following conditions apply:

- `general_profile_idc` is equal to 3 or `general_profile_compatibility_flag[3]` is equal to 1.
- `general_level_idc` is not equal to 255 and represents a level lower than or equal to the specified level.
- `general_tier_flag` represents a tier lower than or equal to the specified tier.

A.3.5 Format range extensions profiles

The following profiles, collectively referred to as the format range extensions profiles, are specified in this clause:

- The Monochrome, Monochrome 10, Monochrome 12 and Monochrome 16 profiles
- The Main 12 profile
- The Main 4:2:2 10 and Main 4:2:2 12 profiles
- The Main 4:4:4, Main 4:4:4 10 and Main 4:4:4 12 profiles
- The Main Intra, Main 10 Intra, Main 12 Intra, Main 4:2:2 10 Intra, Main 4:2:2 12 Intra, Main 4:4:4 Intra, Main 4:4:4 10 Intra, Main 4:4:4 12 Intra and Main 4:4:4 16 Intra profiles
- The Main 4:4:4 Still Picture and Main 4:4:4 16 Still Picture profiles

Bitstreams conforming to the format range extensions profiles shall obey the following constraints:

- The constraints specified in Table A.1 shall apply, in which entries marked with "–" indicate that the table entry does not impose a profile-specific constraint on the corresponding syntax element.

NOTE – For some syntax elements with table entries marked with "–", a constraint may be imposed indirectly – e.g., by semantics constraints that are imposed elsewhere in this Specification when other specified constraints are fulfilled.

- Active VPSs shall have `vps_base_layer_internal_flag` and `vps_base_layer_available_flag` both equal to 1 only.
- Active SPSs for the base layer shall have `separate_colour_plane_flag`, `cabac_bypass_alignment_enabled_flag`, `sps_curr_pic_ref_enabled_flag`, `palette_mode_enabled_flag`, `motion_vector_resolution_control_idc`, and `intra_boundary_filtering_disabled_flag`, when present, equal to 0 only.
- `CtbLog2SizeY` derived according to active SPSs for the base layer shall be in the range of 4 to 6, inclusive.
- Active PPSs for the base layer shall have `residual_adaptive_colour_transform_enabled_flag`, when present, equal to 0 only.
- When an active PPS for the base layer has `tiles_enabled_flag` equal to 1, it shall have `entropy_coding_sync_enabled_flag` equal to 0.
- When an active PPS for the base layer has `tiles_enabled_flag` equal to 1, `ColumnWidthInLumaSamples[i]` shall be greater than or equal to 256 for all values of `i` in the range of 0 to `num_tile_columns_minus1`, inclusive, and `RowHeightInLumaSamples[j]` shall be greater than or equal to 64 for all values of `j` in the range of 0 to `num_tile_rows_minus1`, inclusive.
- In bitstreams conforming to the Main Intra, Main 10 Intra, Main 12 Intra, Main 4:2:2 10 Intra, Main 4:2:2 12 Intra, Main 4:4:4 Intra, Main 4:4:4 10 Intra, Main 4:4:4 12 Intra or Main 4:4:4 16 Intra profiles, all pictures with `nuh_layer_id` equal to 0 shall be IRAP pictures and the output order indicated in the bitstream among these pictures shall be the same as the decoding order.
- The number of times `read_bits(1)` is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing `coding_tree_unit()` data for any CTU shall be less than or equal to $5 * \text{RawCtuBits} / 3$.
- In bitstreams conforming to the Main 4:4:4 Still Picture and Main 4:4:4 16 Still Picture profiles, the following constraints shall apply:
 - The bitstream shall contain only one picture with `nuh_layer_id` equal to 0.
 - Active SPSs for the base layer shall have `sps_max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1]` equal to 0 only.
- In bitstreams conforming to the Monochrome, Monochrome 10, Monochrome 12, Monochrome 16, Main 12, Main 4:2:2 10, Main 4:2:2 12, Main 4:4:4, Main 4:4:4 10, Main 4:4:4 12, Main Intra, Main 10 Intra, Main 12 Intra, Main 4:2:2 10 Intra, Main 4:2:2 12 Intra, Main 4:4:4 Intra, Main 4:4:4 10 Intra, Main 4:4:4 12 Intra or Main 4:4:4 16 Intra profiles, `general_level_idc` and `sub_layer_level_idc[i]` for all values of `i` in active SPSs for the base layer shall not be equal to 255 (which indicates level 8.5).
- The tier and level constraints specified for the Monochrome, Monochrome 10, Monochrome 12, Monochrome 16, Main 12, Main 4:2:2 10, Main 4:2:2 12, Main 4:4:4 10, Main 4:4:4 12, Main Intra, Main 10 Intra, Main 12 Intra, Main 4:2:2 10 Intra, Main 4:2:2 12 Intra, Main 4:4:4 10 Intra, Main 4:4:4 12 Intra or Main 4:4:4 16 Intra profiles in clause A.4, as applicable, shall be fulfilled.

Table A.1 – Allowed values for syntax elements in the format range extensions profiles

Profile for which constraint is specified	chroma_format_idc	bit_depth_luma_minus8 and bit_depth_chroma_minus8	transform_skip_rotation_enabled_flag, transform_skip_context_enabled_flag, implicit_rdpem_enabled_flag, explicit_rdpem_enabled_flag, intra_smoothing_disabled_flag, persistent_rice_adaptation_enabled_flag and log2_max_transform_skip_block_size_minus2	extended_precision_processing_flag	chroma_qp_offset_list_enabled_flag
Monochrome	0	0	0	0	0
Monochrome 10	0	0..2	0	0	0
Monochrome 12	0	0..4	0	0	0
Monochrome 16	0	–	–	–	0
Main 12	0 or 1	0..4	0	0	0
Main 4:2:2 10	0..2	0..2	0	0	–
Main 4:2:2 12	0..2	0..4	0	0	–
Main 4:4:4	–	0	–	0	–
Main 4:4:4 10	–	0..2	–	0	–
Main 4:4:4 12	–	0..4	–	0	–
Main Intra	0 or 1	0	0	0	0
Main 10 Intra	0 or 1	0..2	0	0	0
Main 12 Intra	0 or 1	0..4	0	0	0
Main 4:2:2 10 Intra	0..2	0..2	0	0	–
Main 4:2:2 12 Intra	0..2	0..4	0	0	–
Main 4:4:4 Intra	–	0	–	0	–
Main 4:4:4 10 Intra	–	0..2	–	0	–
Main 4:4:4 12 Intra	–	0..4	–	0	–
Main 4:4:4 16 Intra	–	–	–	–	–
Main 4:4:4 Still Picture	–	0	–	0	–
Main 4:4:4 16 Still Picture	–	–	–	–	–

Conformance of a bitstream to the format range extensions profiles is indicated by `general_profile_idc` being equal to 4 or `general_profile_compatibility_flag[4]` being equal to 1 with the additional indications specified in Table A.2. Conformance of a sub-layer representation with `TemporalId` equal to `i` to the format range extensions profiles is indicated by `sub_layer_profile_idc[i]` being equal to 4 or `sub_layer_profile_compatibility_flag[i][4]` being equal to 1 with the additional indications specified in Table A.2, with each of the syntax elements in Table A.2 being replaced by its `i`-th corresponding sub-layer syntax element.

All other combinations of the syntax elements in Table A.2 with `general_profile_idc` equal to 4 or `general_profile_compatibility_flag[4]` equal to 1 are reserved for future use by ITU-T | ISO/IEC. All other combinations of the *i*-th corresponding sub-layer syntax elements of the syntax elements in Table A.2 with `sub_layer_profile_idc[i]` equal to 4 or `sub_layer_profile_compatibility_flag[i][4]` equal to 1 are reserved for future use by ITU-T | ISO/IEC. Such combinations shall not be present in bitstreams conforming to this Specification. However, decoders conforming to the format range extensions profiles shall allow other combinations as specified below in this clause to occur in the bitstream.

Table A.2 – Bitstream indications for conformance to format range extensions profiles

Profile for which the bitstream indicates conformance	<code>general_max_12bit_constraint_flag</code>	<code>general_max_10bit_constraint_flag</code>	<code>general_max_8bit_constraint_flag</code>	<code>general_max_422chroma_constraint_flag</code>	<code>general_max_420chroma_constraint_flag</code>	<code>general_max_monochrome_constraint_flag</code>	<code>general_intra_constraint_flag</code>	<code>general_one_picture_only_constraint_flag</code>	<code>general_lower_bit_rate_constraint_flag</code>
Monochrome	1	1	1	1	1	1	0	0	1
Monochrome 10	1	1	0	1	1	1	0	0	1
Monochrome 12	1	0	0	1	1	1	0	0	1
Monochrome 16	0	0	0	1	1	1	0	0	1
Main 12	1	0	0	1	1	0	0	0	1
Main 4:2:2 10	1	1	0	1	0	0	0	0	1
Main 4:2:2 12	1	0	0	1	0	0	0	0	1
Main 4:4:4	1	1	1	0	0	0	0	0	1
Main 4:4:4 10	1	1	0	0	0	0	0	0	1
Main 4:4:4 12	1	0	0	0	0	0	0	0	1
Main Intra	1	1	1	1	1	0	1	0	0 or 1
Main 10 Intra	1	1	0	1	1	0	1	0	0 or 1
Main 12 Intra	1	0	0	1	1	0	1	0	0 or 1
Main 4:2:2 10 Intra	1	1	0	1	0	0	1	0	0 or 1
Main 4:2:2 12 Intra	1	0	0	1	0	0	1	0	0 or 1
Main 4:4:4 Intra	1	1	1	0	0	0	1	0	0 or 1
Main 4:4:4 10 Intra	1	1	0	0	0	0	1	0	0 or 1
Main 4:4:4 12 Intra	1	0	0	0	0	0	1	0	0 or 1
Main 4:4:4 16 Intra	0	0	0	0	0	0	1	0	0 or 1
Main 4:4:4 Still Picture	1	1	1	0	0	0	1	1	0 or 1
Main 4:4:4 16 Still Picture	0	0	0	0	0	0	1	1	0 or 1

Decoders conforming to a format range extensions profile at a specific level (identified by a specific value of `general_level_idc`) of a specific tier (identified by a specific value of `general_tier_flag`) shall be capable of decoding all bitstreams and sub-layer representations for which all of the following conditions apply:

- Any of the following conditions apply:
 - The decoder conforms to the Main 12, Main 4:2:2 10, Main 4:2:2 12, Main 4:4:4, Main 4:4:4 10 or Main 4:4:4 12 profile, and the bitstream or sub-layer representation is indicated to conform to the Main profile or the Main Still Picture profile.
 - The decoder conforms to the Main 12, Main 4:2:2 10, Main 4:2:2 12, Main 4:4:4 10 or Main 4:4:4 12 profile, and the bitstream or sub-layer representation is indicated to conform to the Main 10 profile, the Main profile or the Main Still Picture profile.
 - The decoder conforms to the Main Intra, Main 10 Intra, Main 12 Intra, Main 4:2:2 10 Intra, Main 4:2:2 12 Intra, Main 4:4:4 Intra, Main 4:4:4 10 Intra, or Main 4:4:4 12 Intra, Main 4:4:4 16 Intra, Main 4:4:4 Still Picture, or Main 4:4:4 16 Still Picture profile, and the bitstream or sub-layer representation is indicated to conform to the Main Still Picture profile.
 - `general_profile_idc` is equal to 4 or `general_profile_compatibility_flag[4]` is equal to 1 for the bitstream, and the value of each constraint flag listed in Table A.2 is greater than or equal to the value(s) specified in the row of Table A.2 for the format range extensions profile for which the decoder conformance is evaluated.
 - `sub_layer_profile_idc[i]` is equal to 4 or `sub_layer_profile_compatibility_flag[i][4]` is equal to 1 for the sub-layer representation, and the value of each constraint flag listed in Table A.2 is greater than or equal to the value(s) specified in the row of Table A.2 for the format range extensions profile for which the decoder conformance is evaluated, with each of the syntax elements in Table A.2 being replaced by its *i*-th corresponding sub-layer syntax element.
- The bitstream or sub-layer representation is indicated to conform to a level that is not level 8.5 and is lower than or equal to the specified level.
- The bitstream or sub-layer representation is indicated to conform to a tier that is lower than or equal to the specified tier.

For decoders conforming to the Main Intra, Main 10 Intra, Main 12 Intra, Main 4:2:2 10 Intra, Main 4:2:2 12 Intra, Main 4:4:4 Intra, Main 4:4:4 10 Intra, Main 4:4:4 12 Intra, Main 4:4:4 16 Intra, Main 4:4:4 Still Picture, or Main 4:4:4 16 Still picture profile, the application of either or both of the in-loop filters of the in-loop filter process specified in clause 8.7 is optional.

A.3.6 High throughput profiles

The following profiles, collectively referred to as the high throughput profiles, are specified in this clause:

- The High Throughput 4:4:4, High Throughput 4:4:4 10 and High Throughput 4:4:4 14 profiles
- The High Throughput 4:4:4 16 Intra profile

NOTE 1 – For purposes of this terminology, the high-throughput screen content coding extensions profiles specified in clause A.3.8 are not included in the set of profiles that are collectively referred to as the high throughput profiles, although the names of some of the high-throughput screen content coding extensions profiles include the term "High Throughput".

Bitstreams conforming to the high throughput profiles shall obey the following constraints:

- Active VPSs shall have `vps_base_layer_internal_flag` and `vps_base_layer_available_flag` both equal to 1 only.
- Active SPSs for the base layer shall have `separate_colour_plane_flag`, `sps_curr_pic_ref_enabled_flag`, `palette_mode_enabled_flag`, `motion_vector_resolution_control_idc`, and `intra_boundary_filtering_disabled_flag`, when present, equal to 0 only.
- In bitstreams conforming to the High Throughput 4:4:4 profile, active SPSs for the base layer shall have `bit_depth_luma_minus8` equal to 0, `bit_depth_chroma_minus8` equal to 0, `extended_precision_processing_flag` equal to 0, and `cabac_bypass_alignment_enabled_flag` equal to 0 only.
- In bitstreams conforming to the High Throughput 4:4:4 10 profile, active SPSs for the base layer shall have `bit_depth_luma_minus8` less than or equal to 2, `bit_depth_chroma_minus8` less than or equal to 2, `extended_precision_processing_flag` equal to 0, and `cabac_bypass_alignment_enabled_flag` equal to 0 only.
- In bitstreams conforming to the High Throughput 4:4:4 14 profile, active SPSs for the base layer shall have `bit_depth_luma_minus8` less than or equal to 6 and `bit_depth_chroma_minus8` less than or equal to 6.

- In bitstreams conforming to the High Throughput 4:4:4 16 Intra profile, active SPSs for the base layer shall have `cabac_bypass_alignment_enabled_flag` equal to 1 only.
- `CtbLog2SizeY` derived according to active SPSs for the base layer shall be in the range of 4 to 6, inclusive.
- Active PPSs for the base layer shall have `residual_adaptive_colour_transform_enabled_flag`, when present, equal to 0 only.
- In bitstreams conforming to the High Throughput 4:4:4, High Throughput 4:4:4 10, or High Throughput 4:4:4 14 profiles, active PPSs for the base layer shall have `entropy_coding_sync_enabled_flag` equal to 1 only.
 NOTE 2 – Unlike for some other profiles specified in this annex, an active PPS for the base layer for the high throughput profiles may have `tiles_enabled_flag` equal to 1 with `entropy_coding_sync_enabled_flag` equal to 1.
- When an active PPS for the base layer has `tiles_enabled_flag` equal to 1, `ColumnWidthInLumaSamples[i]` shall be greater than or equal to 256 for all values of `i` in the range of 0 to `num_tile_columns_minus1`, inclusive, and `RowHeightInLumaSamples[j]` shall be greater than or equal to 64 for all values of `j` in the range of 0 to `num_tile_rows_minus1`, inclusive.
- In bitstreams conforming to the High Throughput 4:4:4 16 Intra profile, all pictures with `nuh_layer_id` equal to 0 shall be IRAP pictures and the output order indicated in the bitstream among these pictures shall be the same as the decoding order.
- The number of times `read_bits(1)` is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing `coding_tree_unit()` data for any CTU shall be less than or equal to $5 * \text{RawCtuBits} / 3$.
- `general_level_idc` and `sub_layer_level_idc[i]` for all values of `i` in active SPSs for the base layer shall not be equal to 255 (which indicates level 8.5).
- The tier and level constraints specified for the High Throughput 4:4:4, High Throughput 4:4:4 10, High Throughput 4:4:4 14 or High Throughput 4:4:4 16 Intra profile in clause A.4, as applicable, shall be fulfilled.

Conformance of a bitstream to the high throughput profiles is indicated by `general_profile_idc` being equal to 5 or `general_profile_compatibility_flag[5]` being equal to 1 with the additional indications specified in Table A.3. Conformance of a sub-layer representation with `TemporalId` equal to `i` to the high throughput profiles is indicated by `sub_layer_profile_idc[i]` being equal to 5 or `sub_layer_profile_compatibility_flag[i][5]` being equal to 1 with the additional indications specified in Table A.3, with each of the syntax elements in Table A.3 being replaced by its `i`-th corresponding sub-layer syntax element.

All other combinations of the syntax elements in Table A.3 with `general_profile_idc` equal to 5 or `general_profile_compatibility_flag[5]` equal to 1 are reserved for future use by ITU-T | ISO/IEC. All other combinations of the `i`-th corresponding sub-layer syntax elements of the syntax elements in Table A.3 with `sub_layer_profile_idc[i]` equal to 5 or `sub_layer_profile_compatibility_flag[i][5]` equal to 1 are reserved for future use by ITU-T | ISO/IEC. Such combinations shall not be present in bitstreams conforming to this Specification. However, decoders conforming to the format range extensions profiles shall allow other combinations as specified below in this clause to occur in the bitstream.

Table A.3 – Bitstream indications for conformance to high throughput profiles

Profile for which the bitstream indicates conformance	general_lower_bit_rate_constraint_flag	general_one_picture_only_constraint_flag	general_intra_constraint_flag	general_max_monochrome_constraint_flag	general_max_420chroma_constraint_flag	general_max_422chroma_constraint_flag	general_max_8bit_constraint_flag	general_max_10bit_constraint_flag	general_max_12bit_constraint_flag	general_max_14bit_constraint_flag
High Throughput 4:4:4	1	1	1	1	0	0	0	0	0	1
High Throughput 4:4:4 10	1	1	1	0	0	0	0	0	0	1
High Throughput 4:4:4 14	1	0	0	0	0	0	0	0	0	1
High Throughput 4:4:4 16 Intra	0	0	0	0	0	0	0	0	1	0 or 1

Decoders conforming to a high throughput profile at a specific level (identified by a specific value of `general_level_idc`) of a specific tier (identified by a specific value of `general_tier_flag`) shall be capable of decoding all bitstreams or sub-layer representations for which all of the following conditions apply:

- Any of the following conditions apply:
 - `general_profile_idc` is equal to 5 or `general_profile_compatibility_flag[5]` is equal to 1 for the bitstream and the value of each constraint flag listed in Table A.3 is greater than or equal to the value(s) specified in the row of Table A.3 for the high throughput profile for which the decoder conformance is evaluated.
 - `sub_layer_profile_idc[i]` is equal to 5 or `sub_layer_profile_compatibility_flag[i][5]` is equal to 1 for the sub-layer representation, and the value of each constraint flag listed in Table A.3 is greater than or equal to the value(s) specified in the row of Table A.3 for the high throughput profile for which the decoder conformance is evaluated, with each of the syntax elements in Table A.3 being replaced by its *i*-th corresponding sub-layer syntax element.
- The bitstream or sub-layer representation is indicated to conform to a level that is not level 8.5 and is lower than or equal to the specified level.
- The bitstream or sub-layer representation is indicated to conform to a tier that is lower than or equal to the specified tier.

For decoders conforming to the High Throughput 4:4:4 16 Intra profile, the application of either or both of the in-loop filters of the in-loop filter process specified in clause 8.7 is optional.

A.3.7 Screen content coding extensions profiles

The following profiles, collectively referred to as the screen content coding extensions profiles, are specified in this clause:

- The Screen-Extended Main and Screen-Extended Main 10 profiles
- The Screen-Extended Main 4:4:4 and Screen-Extended Main 4:4:4 10 profiles

NOTE – For purposes of this terminology, the high throughput screen content coding extensions profiles specified in clause A.3.8 are not included in the set of profiles that are collectively referred to as the screen content coding extensions profiles, although the names of some of the high throughput screen content coding extensions profiles include the term "Screen-Extended".

Bitstreams conforming to the screen content coding extensions profiles shall obey the following constraints:

- The constraints specified in Table A.4 shall apply, in which entries marked with "-" indicate that the table entry does not impose a profile-specific constraint on the corresponding syntax element.

- Active VPSs shall have `vps_base_layer_internal_flag` and `vps_base_layer_available_flag` both equal to 1 only.
- Active SPSs for the base layer shall have `separate_colour_plane_flag`, when present, equal to 0 only.
- `CtbLog2SizeY` derived according to active SPSs for the base layer shall be in the range of 4 to 6, inclusive.
- When an active SPS for the base layer has `palette_mode_enabled_flag` equal to 1, `palette_max_size` shall be less than or equal to 64 and `PaletteMaxPredictorSize` shall be less than or equal to 128.
- In bitstreams conforming to the Screen-Extended Main, Screen-Extended Main 10, Screen-Extended Main 4:4:4, or Screen-Extended Main 4:4:4 10 profiles, active SPSs for the base layer shall have `extended_precision_processing_flag`, and `cabac_bypass_alignment_enabled_flag`, when present, equal to 0 only.
- In bitstreams conforming to the Screen-Extended Main or Screen-Extended Main 10 profiles, when an active PPS for the base layer has `tiles_enabled_flag` equal to 1, it shall have `entropy_coding_sync_enabled_flag` equal to 0.
- When an active PPS for the base layer has `tiles_enabled_flag` equal to 1, `ColumnWidthInLumaSamples[i]` shall be greater than or equal to 256 for all values of `i` in the range of 0 to `num_tile_columns_minus1`, inclusive, and `RowHeightInLumaSamples[j]` shall be greater than or equal to 64 for all values of `j` in the range of 0 to `num_tile_rows_minus1`, inclusive.
- The number of times `read_bits(1)` is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing `coding_tree_unit()` data for any CTU shall be less than or equal to $5 * \text{RawCtuBits} / 3$.
- `general_level_idc` and `sub_layer_level_idc[i]` for all values of `i` in active SPSs for the base layer shall not be equal to 255 (which indicates level 8.5).
- The tier and level constraints specified for the Screen-Extended Main, Screen-Extended Main 10, Screen-Extended Main 4:4:4 or Screen-Extended Main 4:4:4 10 profiles in clause A.4, as applicable, shall be fulfilled.

Table A.4 – Allowed values for syntax elements in the screen content coding extensions profiles

Profile for which constraint is specified	<code>chroma_format_idc</code>	<code>bit_depth_luma_minus8</code> and <code>bit_depth_chroma_minus8</code>
Screen-Extended Main	1	0
Screen-Extended Main 10	1	0..2
Screen-Extended Main 4:4:4	0, 1, or 3	0
Screen-Extended Main 4:4:4 10	0, 1, or 3	0..2

Conformance of a bitstream to the screen content coding extensions profiles is indicated by `general_profile_idc` being equal to 9 or `general_profile_compatibility_flag[9]` being equal to 1 with the additional indications specified in Table A.5. Conformance of a sub-layer representation with `TemporalId` equal to `i` to the screen content coding extensions profiles is indicated by `sub_layer_profile_idc[i]` being equal to 9 or `sub_layer_profile_compatibility_flag[i][9]` being equal to 1 with the additional indications specified in Table A.5, with each of the syntax elements in Table A.5 being replaced by its `i`-th corresponding sub-layer syntax element.

All other combinations of the syntax elements in Table A.5 with `general_profile_idc` equal to 9 or `general_profile_compatibility_flag[9]` equal to 1 are reserved for future use by ITU-T | ISO/IEC. All other combinations of the *i*-th corresponding sub-layer syntax elements of the syntax elements in Table A.5 with `sub_layer_profile_idc[i]` equal to 9 or `sub_layer_profile_compatibility_flag[i][9]` equal to 1 are reserved for future use by ITU-T | ISO/IEC. Such combinations shall not be present in bitstreams conforming to this Specification. However, decoders conforming to the screen content coding extensions profiles shall allow other combinations as specified below in this clause to occur in the bitstream.

Table A.5 – Bitstream indications for conformance to screen content coding extensions profiles

Profile for which the bitstream indicates conformance	<code>general_max_14bit_constraint_flag</code>	<code>general_max_12bit_constraint_flag</code>	<code>general_max_10bit_constraint_flag</code>	<code>general_max_8bit_constraint_flag</code>	<code>general_max_422chroma_constraint_flag</code>	<code>general_max_420chroma_constraint_flag</code>	<code>general_max_monochrome_constraint_flag</code>	<code>general_intra_constraint_flag</code>	<code>general_one_picture_only_constraint_flag</code>	<code>general_lower_bit_rate_constraint_flag</code>
Screen-Extended Main	1	1	1	1	1	1	0	0	0	1
Screen-Extended Main 10	1	1	1	0	1	1	0	0	0	1
Screen-Extended Main 4:4:4	1	1	1	1	0	0	0	0	0	1
Screen-Extended Main 4:4:4 10	1	1	1	0	0	0	0	0	0	1

Decoders conforming to a screen content coding extensions profile at a specific level (identified by a specific value of `general_level_idc`) of a specific tier (identified by a specific value of `general_tier_flag`) shall be capable of decoding all bitstreams and sub-layer representations for which all of the following conditions apply:

- Any of the following conditions apply:
 - The bitstream or sub-layer representation is indicated to conform to the Main, Main Still Picture, or Monochrome profile.
 - The decoder conforms to the Screen-Extended Main 10 or Screen-Extended Main 4:4:4 10 profile, and the bitstream or sub-layer representation is indicated to conform to the Main 10 profile.
 - The decoder conforms to the Screen-Extended Main 4:4:4 or Screen-Extended Main 4:4:4 10 profile, and the bitstream or sub-layer representation is indicated to conform to the Main 4:4:4 profile.
 - The decoder conforms to the Screen-Extended Main 4:4:4 10 profile, and the bitstream or sub-layer representation is indicated to conform to the Main 4:4:4 10 profile.
 - `general_profile_idc` is equal to 4 or `general_profile_compatibility_flag[4]` is equal to 1 or `general_profile_idc` is equal to 9 or `general_profile_compatibility_flag[9]` is equal to 1 for the bitstream, and the value of each constraint flag listed in Table A.5 is greater than or equal to the value(s) specified in the row of Table A.5 for the screen content coding extensions profile for which the decoder conformance is evaluated, and `general_max_422chroma_constraint_flag` is equal to `general_max_420chroma_constraint_flag`.
 - `sub_layer_profile_idc[i]` is equal to 4 or `sub_layer_profile_compatibility_flag[i][4]` is equal to 1 or `sub_layer_profile_idc[i]` is equal to 9 or `sub_layer_profile_compatibility_flag[i][9]` is equal to 1 for the sub-layer representation, and the value of each constraint flag listed in Table A.5 is greater than or equal to the value(s) specified in the row of Table A.5 for the screen content coding extensions profile for which the decoder conformance is evaluated, and `general_max_422chroma_constraint_flag` is equal to `general_max_420chroma_constraint_flag`, with each of the syntax elements in Table A.5 being replaced by its *i*-th corresponding sub-layer syntax element.

- The bitstream or sub-layer representation is indicated to conform to a level that is not level 8.5 and is lower than or equal to the specified level.
- The bitstream or sub-layer representation is indicated to conform to a tier that is lower than or equal to the specified tier.

A.3.8 High throughput screen content coding extensions profiles

The following profiles, collectively referred to as the high throughput screen content coding extensions profiles, are specified in this clause:

- The Screen-Extended High Throughput 4:4:4, Screen-Extended High Throughput 4:4:4 10, and Screen-Extended High Throughput 14 profiles

Bitstreams conforming to the screen content coding extensions profiles shall obey the following constraints:

- The constraints specified in Table A.6 shall apply, in which entries marked with "-" indicate that the table entry does not impose a profile-specific constraint on the corresponding syntax element.
- Active VPSs shall have `vps_base_layer_internal_flag` and `vps_base_layer_available_flag` both equal to 1 only.
- Active SPSs for the base layer shall have `separate_colour_plane_flag`, when present, equal to 0 only.
- `CtbLog2SizeY` derived according to active SPSs for the base layer shall be in the range of 4 to 6, inclusive.
- When an active SPS for the base layer has `palette_mode_enabled_flag` equal to 1, `palette_max_size` shall be less than or equal to 64 and `PaletteMaxPredictorSize` shall be less than or equal to 128.
- Active SPSs for the base layer shall have `extended_precision_processing_flag`, and `cabac_bypass_alignment_enabled_flag`, when present, equal to 0 only.
- Active PPSs for the base layer shall have `entropy_coding_sync_enabled_flag` equal to 1 only.
NOTE – Unlike for some other profiles specified in this annex, an active PPS for the base layer for Screen-Extended High Throughput 4:4:4, Screen-Extended High Throughput 4:4:4 10, or Screen-Extended High Throughput 4:4:4 14 profiles may have `tiles_enabled_flag` equal to 1 with `entropy_coding_sync_enabled_flag` equal to 1.
- When an active PPS for the base layer has `tiles_enabled_flag` equal to 1, `ColumnWidthInLumaSamples[i]` shall be greater than or equal to 256 for all values of `i` in the range of 0 to `num_tile_columns_minus1`, inclusive, and `RowHeightInLumaSamples[j]` shall be greater than or equal to 64 for all values of `j` in the range of 0 to `num_tile_rows_minus1`, inclusive.
- The number of times `read_bits(1)` is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing `coding_tree_unit()` data for any CTU shall be less than or equal to $5 * \text{RawCtuBits} / 3$.
- `general_level_idc` and `sub_layer_level_idc[i]` for all values of `i` in active SPSs for the base layer shall not be equal to 255 (which indicates level 8.5).
- The tier and level constraints specified for the Screen-Extended High Throughput 4:4:4, Screen-Extended High Throughput 4:4:4 10, and Screen-Extended High Throughput 14 profiles in clause A.4, as applicable, shall be fulfilled.

Table A.6 – Allowed values for syntax elements in the high throughput screen content coding extensions profiles

Profile for which constraint is specified	chroma_format_idc	bit_depth_luma_minus8 and bit_depth_chroma_minus8
Screen-Extended High Throughput 4:4:4	–	0
Screen-Extended High Throughput 4:4:4 10	–	0..2
Screen-Extended High Throughput 4:4:4 14	–	0..6

Conformance of a bitstream to the high throughput screen content coding extensions profiles is indicated by `general_profile_idc` being equal to 11 or `general_profile_compatibility_flag[11]` being equal to 1 with the additional indications specified in Table A.7. Conformance of a sub-layer representation with `TemporalId` equal to `i` to the screen content coding extensions profiles is indicated by `sub_layer_profile_idc[i]` being equal to 11 or `sub_layer_profile_compatibility_flag[i][11]` being equal to 1 with the additional indications specified in Table A.7, with each of the syntax elements in Table A.7 being replaced by its `i`-th corresponding sub-layer syntax element.

All other combinations of the syntax elements in Table A.7 with `general_profile_idc` equal to 11 or `general_profile_compatibility_flag[11]` equal to 1 are reserved for future use by ITU-T | ISO/IEC. All other combinations of the `i`-th corresponding sub-layer syntax elements of the syntax elements in Table A.7 with `sub_layer_profile_idc[i]` equal to 11 or `sub_layer_profile_compatibility_flag[i][11]` equal to 1 are reserved for future use by ITU-T | ISO/IEC. Such combinations shall not be present in bitstreams conforming to this Specification. However, decoders conforming to the screen content coding extensions profiles shall allow other combinations as specified below in this clause to occur in the bitstream.

Table A.7 – Bitstream indications for conformance to high throughput screen content coding extensions profiles

Profile for which the bitstream indicates conformance	general_max_14bit_constraint_flag	general_max_12bit_constraint_flag	general_max_10bit_constraint_flag	general_max_8bit_constraint_flag	general_max_422chroma_constraint_flag	general_max_420chroma_constraint_flag	general_max_monochrome_constraint_flag	general_intra_constraint_flag	general_one_picture_only_constraint_flag	general_lower_bit_rate_constraint_flag
Screen-Extended High Throughput 4:4:4	1	1	1	1	0	0	0	0	0	1
Screen-Extended High Throughput 4:4:4 10	1	1	1	0	0	0	0	0	0	1
Screen-Extended High Throughput 4:4:4 14	1	0	0	0	0	0	0	0	0	1

Decoders conforming to a high throughput screen content coding extensions profile at a specific level (identified by a specific value of `general_level_idc`) of a specific tier (identified by a specific value of `general_tier_flag`) shall be capable of decoding all bitstreams and sub-layer representations for which all of the following conditions apply:

- Any of the following conditions apply:
 - The bitstream or sub-layer representation is indicated to conform to the Main, Main Still Picture, or Monochrome profile.
 - The bitstream or sub-layer representation is indicated to conform to the High Throughput 4:4:4 profile.
 - The decoder conforms to the Screen-Extended High Throughput 4:4:4 10 or Screen-Extended High Throughput 4:4:4 14 profile, and the bitstream or sub-layer representation is indicated to conform to the High Throughput 4:4:4 10 profile.
 - The decoder conforms to the Screen-Extended High Throughput 4:4:4 14 profile, and the bitstream or sub-layer representation is indicated to conform to the High Throughput 4:4:4 14 profile.
 - `general_profile_idc` is equal to 4 or `general_profile_compatibility_flag[4]` is equal to 1 or `general_profile_idc` is equal to 11 or `general_profile_compatibility_flag[11]` is equal to 1 for the bitstream, and the value of each constraint flag listed in Table A.7 is greater than or equal to the value(s) specified in the row of Table A.7 for the screen content coding extensions profile for which the decoder conformance is evaluated, and `general_max_422chroma_constraint_flag` is equal to `general_max_420chroma_constraint_flag`.
 - `sub_layer_profile_idc[i]` is equal to 4 or `sub_layer_profile_compatibility_flag[i][4]` is equal to 1 or `sub_layer_profile_idc[i]` is equal to 11 or `sub_layer_profile_compatibility_flag[i][11]` is equal to 1 for the sub-layer representation, and the value of each constraint flag listed in Table A.7 is greater than or equal to the value(s) specified in the row of Table A.7 for the screen content coding extensions profile for which the decoder conformance is evaluated, and `general_max_422chroma_constraint_flag` is equal to `general_max_420chroma_constraint_flag`, with each of the syntax elements in Table A.7 being replaced by its *i*-th corresponding sub-layer syntax element, respectively.
- The bitstream or sub-layer representation is indicated to conform to a level that is not level 8.5 and is lower than or equal to the specified level.
- The bitstream or sub-layer representation is indicated to conform to a tier that is lower than or equal to the specified tier.

A.4 Tiers and levels

A.4.1 General tier and level limits

For purposes of comparison of tier capabilities, the tier with `general_tier_flag` or `sub_layer_tier_flag[i]` equal to 0 is considered to be a lower tier than the tier with `general_tier_flag` or `sub_layer_tier_flag[i]` equal to 1.

For purposes of comparison of level capabilities, a particular level of a specific tier is considered to be a lower level than some other level of the same tier when the value of the `general_level_idc` or `sub_layer_level_idc[i]` of the particular level is less than that of the other level.

The following is specified for expressing the constraints in this annex:

- Let access unit *n* be the *n*-th access unit in decoding order, with the first access unit being access unit 0 (i.e., the 0-th access unit).
- Let picture *n* be the coded picture or the corresponding decoded picture of access unit *n*.

When the specified level is not level 8.5, bitstreams conforming to a profile at a specified tier and level shall obey the following constraints for each bitstream conformance test as specified in Annex C:

- a) `PicSizeInSamplesY` shall be less than or equal to `MaxLumaPs`, where `MaxLumaPs` is specified in Table A.8.
- b) The value of `pic_width_in_luma_samples` shall be less than or equal to $\text{Sqrt}(\text{MaxLumaPs} * 8)$.
- c) The value of `pic_height_in_luma_samples` shall be less than or equal to $\text{Sqrt}(\text{MaxLumaPs} * 8)$.
- d) For level 5 and higher levels, the value of `CtbSizeY` shall be equal to 32 or 64.
- e) The value of `NumPicTotalCurr` shall be less than or equal to 8.
- f) The value of `num_tile_columns_minus1` shall be less than `MaxTileCols` and `num_tile_rows_minus1` shall be less than `MaxTileRows`, where `MaxTileCols` and `MaxTileRows` are specified in Table A.8.
- g) For the VCL HRD parameters, `CpbSize[i]` shall be less than or equal to `CpbVclFactor * MaxCPB` for at least one value of *i* in the range of 0 to `cpb_cnt_minus1[HighestTid]`, inclusive, where `CpbSize[i]` is specified in clause E.3.3 based on parameters selected as specified in clause C.1, `CpbVclFactor` is specified in Table A.10, and `MaxCPB` is specified in Table A.8 in units of `CpbVclFactor` bits.
- h) For the NAL HRD parameters, `CpbSize[i]` shall be less than or equal to `CpbNalFactor * MaxCPB` for at least one value of *i* in the range of 0 to `cpb_cnt_minus1[HighestTid]`, inclusive, where `CpbSize[i]` is specified in clause E.3.3 based on parameters selected as specified in clause C.1, `CpbNalFactor` is specified in Table A.10, and `MaxCPB` is specified in Table A.8 in units of `CpbNalFactor` bits.

Table A.8 specifies the limits for each level of each tier for levels other than level 8.5.

A tier and level to which a bitstream conforms are indicated by the syntax elements `general_tier_flag` and `general_level_idc`, and a tier and level to which a sub-layer representation conforms are indicated by the syntax elements `sub_layer_tier_flag[i]` and `sub_layer_level_idc[i]`, as follows:

- If the specified level is not level 8.5, `general_tier_flag` or `sub_layer_tier_flag[i]` equal to 0 indicates conformance to the Main tier, `general_tier_flag` or `sub_layer_tier_flag[i]` equal to 1 indicates conformance to the High tier, according to the tier constraints specified in Table A.8 and `general_tier_flag` and `sub_layer_tier_flag[i]` shall be equal to 0 for levels below level 4 (corresponding to the entries in Table A.8 marked with "-"). Otherwise (the specified level is level 8.5), it is a requirement of bitstream conformance that `general_tier_flag` and `sub_layer_tier_flag[i]` shall be equal to 1 and the value 0 for `general_tier_flag` and `sub_layer_tier_flag[i]` is reserved for future use by ITU-T | ISO/IEC and decoders shall ignore the value of `general_tier_flag` and `sub_layer_tier_flag[i]`.
- `general_level_idc` and `sub_layer_level_idc[i]` shall be set equal to a value of 30 times the level number specified in Table A.8.

Table A.8 – General tier and level limits

Level	Max luma picture size MaxLumaPs (samples)	Max CPB size MaxCPB (CpbVclFactor or CpbNalFactor bits)		Max slice segments per picture MaxSliceSegmentsPerPicture	Max # of tile rows MaxTileRows	Max # of tile columns MaxTileCols
		Main tier	High tier			
1	36 864	350	-	16	1	1
2	122 880	1 500	-	16	1	1
2.1	245 760	3 000	-	20	1	1
3	552 960	6 000	-	30	2	2
3.1	983 040	10 000	-	40	3	3
4	2 228 224	12 000	30 000	75	5	5
4.1	2 228 224	20 000	50 000	75	5	5
5	8 912 896	25 000	100 000	200	11	10
5.1	8 912 896	40 000	160 000	200	11	10
5.2	8 912 896	60 000	240 000	200	11	10
6	35 651 584	60 000	240 000	600	22	20
6.1	35 651 584	120 000	480 000	600	22	20
6.2	35 651 584	240 000	800 000	600	22	20

A.4.2 Profile-specific level limits for the video profiles

NOTE – The term "video profiles", as used in this clause, refers to those profiles that are not still picture profiles. The still picture profiles include the Main Still Picture, Main 10 Still Picture, Main 4:4:4 Still Picture, and Main 4:4:4 16 Still Picture profiles.

The following is specified for expressing the constraints in this annex:

- Let the variable fR be set equal to $1 \div 300$.

The variable HbrFactor is defined as follows:

- If the bitstream or sub-layer representation is indicated to conform to the Main profile or the Main 10 profile, HbrFactor is set equal to 1.
- Otherwise, if the bitstream or sub-layer representation is indicated to conform to the High Throughput 4:4:4, High Throughput 4:4:4 10, High Throughput 4:4:4 14, Screen-Extended High Throughput 4:4:4, Screen-Extended High Throughput 4:4:4 10, or Screen-Extended High Throughput 4:4:4 14 profile, HbrFactor is set equal to 6.
- Otherwise, if the bitstream or sub-layer representation is indicated to conform to the High Throughput 4:4:4 16 Intra profile, HbrFactor is set equal to $24 - (12 * \text{general_lower_bit_rate_constraint_flag})$ or $24 - (12 * \text{sub_layer_lower_bit_rate_constraint_flag}[i])$.
- Otherwise, HbrFactor is set equal to $2 - \text{general_lower_bit_rate_constraint_flag}$ or $2 - \text{sub_layer_lower_bit_rate_constraint_flag}[i]$.

The variable BrVclFactor, which represents the VCL bit rate scale factor, is set equal to $\text{CpbVclFactor} * \text{HbrFactor}$.

The variable BrNalFactor, which represents the NAL bit rate scale factor, is set equal to $\text{CpbNalFactor} * \text{HbrFactor}$.

The variable MinCr is set equal to $\text{MinCrBase} * \text{MinCrScaleFactor} \div \text{HbrFactor}$.

When the specified level is not level 8.5, the value of $\text{sps_max_dec_pic_buffering_minus1}[\text{HighestTid}] + 1$ shall be less than or equal to MaxDpbSize, which is derived as follows:

$$\text{if(PicSizeInSamplesY } \leq (\text{MaxLumaPs} \gg 2)) \\ \text{MaxDpbSize} = \text{Min}(4 * \text{maxDpbPicBuf}, 16)$$

```

else if( PicSizeInSamplesY <= ( MaxLumaPs >> 1 ) )
    MaxDpbSize = Min( 2 * maxDpbPicBuf, 16 )
else if( PicSizeInSamplesY <= ( ( 3 * MaxLumaPs ) >> 2 ) )
    MaxDpbSize = Min( ( 4 * maxDpbPicBuf ) / 3, 16 )
else
    MaxDpbSize = maxDpbPicBuf

```

(A-2)

where MaxLumaPs is specified in Table A.8, and maxDpbPicBuf is equal to 6 for all profiles where the value of sps_curr_pic_ref_enabled_flag is required to be equal to 0 and 7 for all profiles where the value of sps_curr_pic_ref_enabled_flag is not required to be equal to 0.

Bitstreams and sub-layer representations conforming to the Monochrome, Monochrome 10, Monochrome 12, Monochrome 16, Main, Main 10, Main 12, Main 4:2:2 10, Main 4:2:2 12, Main 4:4:4 10, Main 4:4:4 12, Main Intra, Main 10 Intra, Main 12 Intra, Main 4:2:2 10 Intra, Main 4:2:2 12 Intra, Main 4:4:4 10 Intra, Main 4:4:4 12 Intra, Main 4:4:4 16 Intra High Throughput 4:4:4, High Throughput 4:4:4 10, High Throughput 4:4:4 14, High Throughput 4:4:4 16 Intra, Screen-Extended Main, Screen-Extended Main 10, Screen-Extended Main 4:4:4, Screen-Extended Main 4:4:4 10, Screen-Extended High Throughput 4:4:4, Screen-Extended High Throughput 4:4:4 10, or Screen-Extended High Throughput 4:4:4 14 profile at a specified tier and level shall obey the following constraints for each bitstream conformance test as specified in Annex C:

- a) The nominal removal time of access unit n (with n greater than 0) from the CPB, as specified in clause C.2.3, shall satisfy the constraint that $AuNominalRemovalTime[n] - AuCpbRemovalTime[n - 1]$ is greater than or equal to $\text{Max}(PicSizeInSamplesY \div MaxLumaSr, fR)$ for the value of $PicSizeInSamplesY$ of picture $n - 1$, where $MaxLumaSr$ is the value specified in Table A.9 that applies to picture $n - 1$.
- b) The difference between consecutive output times of pictures from the DPB, as specified in clause C.3.3, shall satisfy the constraint that $DpbOutputInterval[n]$ is greater than or equal to $\text{Max}(PicSizeInSamplesY \div MaxLumaSr, fR)$ for the value of $PicSizeInSamplesY$ of picture n , where $MaxLumaSr$ is the value specified in Table A.9 for picture n , provided that picture n is a picture that is output and is not the last picture of the bitstream that is output.
- c) The removal time of access unit 0 shall satisfy the constraint that the number of slice segments in picture 0 is less than or equal to $\text{Min}(\text{Max}(1, MaxSliceSegmentsPerPicture * MaxLumaSr / MaxLumaPs * (AuCpbRemovalTime[0] - AuNominalRemovalTime[0]) + MaxSliceSegmentsPerPicture * PicSizeInSamplesY / MaxLumaPs), MaxSliceSegmentsPerPicture)$, for the value of $PicSizeInSamplesY$ of picture 0, where $MaxSliceSegmentsPerPicture$, $MaxLumaPs$ and $MaxLumaSr$ are the values specified in Table A.8 and Table A.9, respectively, that apply to picture 0.
- d) The difference between consecutive CPB removal times of access units n and $n - 1$ (with n greater than 0) shall satisfy the constraint that the number of slice segments in picture n is less than or equal to $\text{Min}(\text{Max}(1, MaxSliceSegmentsPerPicture * MaxLumaSr / MaxLumaPs * (AuCpbRemovalTime[n] - AuCpbRemovalTime[n - 1])), MaxSliceSegmentsPerPicture)$, where $MaxSliceSegmentsPerPicture$, $MaxLumaPs$ and $MaxLumaSr$ are the values specified in Table A.8 and Table A.9 that apply to picture n .
- e) For the VCL HRD parameters, $BitRate[i]$ shall be less than or equal to $BrVclFactor * MaxBR$ for at least one value of i in the range of 0 to $cpb_cnt_minus1[HighestTid]$, inclusive, where $BitRate[i]$ is specified in clause E.3.3 based on parameters selected as specified in clause C.1 and $MaxBR$ is specified in Table A.9 in units of $BrVclFactor$ bits/s.
- f) For the NAL HRD parameters, $BitRate[i]$ shall be less than or equal to $BrNalFactor * MaxBR$ for at least one value of i in the range of 0 to $cpb_cnt_minus1[HighestTid]$, inclusive, where $BitRate[i]$ is specified in clause E.3.3 based on parameters selected as specified in clause C.1 and $MaxBR$ is specified in Table A.9 in units of $BrNalFactor$ bits/s.
- g) The sum of the $NumBytesInNalUnit$ variables for access unit 0 shall be less than or equal to $\text{FormatCapabilityFactor} * (\text{Max}(PicSizeInSamplesY, fR * MaxLumaSr) + MaxLumaSr * (AuCpbRemovalTime[0] - AuNominalRemovalTime[0])) \div MinCr$ for the value of $PicSizeInSamplesY$ of picture 0, where $MaxLumaSr$ and $FormatCapabilityFactor$ are the values specified in Table A.9 and Table A.10, respectively, that apply to picture 0.
- h) The sum of the $NumBytesInNalUnit$ variables for access unit n (with n greater than 0) shall be less than or equal to $\text{FormatCapabilityFactor} * MaxLumaSr * (AuCpbRemovalTime[n] - AuCpbRemovalTime[n - 1]) \div MinCr$, where $MaxLumaSr$ and $FormatCapabilityFactor$ are the values specified in Table A.9 and Table A.10, respectively, that apply to picture n .
- i) The removal time of access unit 0 shall satisfy the constraint that the number of tiles in picture 0 is less than or equal to $\text{Min}(\text{Max}(1, MaxTileCols * MaxTileRows * 120 * (AuCpbRemovalTime[0] -$

$AuNominalRemovalTime[0]) + MaxTileCols * MaxTileRows * PicSizeInSamplesY / MaxLumaPs$), $MaxTileCols * MaxTileRows$), for the value of $PicSizeInSamplesY$ of picture 0, where $MaxTileCols$ and $MaxTileRows$ are the values specified in Table A.8 that apply to picture 0.

- j) The difference between consecutive CPB removal times of access units n and $n - 1$ (with n greater than 0) shall satisfy the constraint that the number of tiles in picture n is less than or equal to $Min(Max(1, MaxTileCols * MaxTileRows * 120 * (AuCpbRemovalTime[n] - AuCpbRemovalTime[n - 1])), MaxTileCols * MaxTileRows)$, where $MaxTileCols$ and $MaxTileRows$ are the values specified in Table A.8 that apply to picture n .

k)

Table A.9 – Tier and level limits for the video profiles

Level	Max luma sample rate MaxLumaSr (samples/sec)	Max bit rate MaxBR (BrVclFactor or BrNalFactor bits/s)		Min compression ratio MinCBase	
		Main tier	High tier	Main tier	High tier
1	552 960	128	-	2	2
2	3 686 400	1 500	-	2	2
2.1	7 372 800	3 000	-	2	2
3	16 588 800	6 000	-	2	2
3.1	33 177 600	10 000	-	2	2
4	66 846 720	12 000	30 000	4	4
4.1	133 693 440	20 000	50 000	4	4
5	267 386 880	25 000	100 000	6	4
5.1	534 773 760	40 000	160 000	8	4
5.2	1 069 547 520	60 000	240 000	8	4
6	1 069 547 520	60 000	240 000	8	4
6.1	2 139 095 040	120 000	480 000	8	4
6.2	4 278 190 080	240 000	800 000	6	4

Table A.10 – Specification of CpbVclFactor, CpbNalFactor, FormatCapabilityFactor and MinCrScaleFactor

Profile	CpbVclFactor	CpbNalFactor	FormatCapabilityFactor	MinCrScaleFactor
Monochrome	667	733	1.000	1.0
Monochrome 10	833	917	1.250	1.0
Monochrome 12	1 000	1 100	1.500	1.0
Monochrome 16	1 333	1 467	2.000	1.0
Main	1 000	1 100	1.500	1.0
Screen-Extended Main	1 000	1 100	1.500	1.0
Main 10	1 000	1 100	1.875	1.0
Screen-Extended Main 10	1 000	1 100	1.875	1.0
Main 12	1 500	1 650	2.250	1.0
Main Still Picture	1 000	1 100	1.500	1.0
Main 10 Still Picture	1 000	1 100	1.875	1.0
Main 4:2:2 10	1 667	1 833	2.500	0.5
Main 4:2:2 12	2 000	2 200	3.000	0.5
Main 4:4:4	2 000	2 200	3.000	0.5
High Throughput 4:4:4	2 000	2 200	3.000	0.5
Screen-Extended Main 4:4:4	2 000	2 200	3.000	0.5
Screen-Extended High Throughput 4:4:4	2 000	2 200	3.000	0.5
Main 4:4:4 10	2 500	2 750	3.750	0.5
High Throughput 4:4:4 10	2 500	2 750	3.750	0.5
Screen-Extended Main 4:4:4 10	2 500	2 750	3.750	0.5
Screen-Extended High Throughput 4:4:4 10	2 500	2 750	3.750	0.5
Main 4:4:4 12	3 000	3 300	4.500	0.5
High Throughput 4:4:4 14	3 500	3 850	5.250	0.5
Screen-Extended High Throughput 4:4:4 14	3 500	3 850	5.250	0.5
Main Intra	1 000	1 100	1.500	1.0
Main 10 Intra	1 000	1 100	1.875	1.0
Main 12 Intra	1 500	1 650	2.250	1.0
Main 4:2:2 10 Intra	1 667	1 833	2.500	0.5
Main 4:2:2 12 Intra	2 000	2 200	3.000	0.5
Main 4:4:4 Intra	2 000	2 200	3.000	0.5
Main 4:4:4 10 Intra	2 500	2 750	3.750	0.5
Main 4:4:4 12 Intra	3 000	3 300	4.500	0.5
Main 4:4:4 16 Intra	4 000	4 400	6.000	0.5
Main 4:4:4 Still Picture	2 000	2 200	3.000	0.5
Main 4:4:4 16 Still Picture	4 000	4 400	6.000	0.5
High Throughput 4:4:4 16 Intra	4 000	4 400	6.000	0.5

Informative clause A.4.3 shows the effect of these limits on picture rates for several example picture formats.

A.4.3 Effect of level limits on picture rate for the video profiles (informative)

This clause does not form an integral part of this Specification.

Informative Tables A.11 and A.12 provide examples of maximum picture rates for the Monochrome, Monochrome 10, Monochrome 12, Monochrome 16, Main, Main 10, Main 12, Main 4:2:2 10, Main 4:2:2 12, Main 4:4:4, Main 4:4:4 10, Main 4:4:4 12, Main Intra, Main 10 Intra, Main 12 Intra, Main 4:2:2 10 Intra, Main 4:2:2 12 Intra, Main 4:4:4 Intra, Main 4:4:4 10 Intra, Main 4:4:4 12 Intra, Main 4:4:4 16 Intra and High Throughput 4:4:4 16 Intra profiles for various picture formats when MinCbSizeY is equal to 64.

**Table A.11 – Maximum picture rates (pictures per second) at level 1 to 4.1 for some example picture sizes
when MinCbSizeY is equal to 64**

Level:				1	2	2.1	3	3.1	4	4.1
Max luma picture size (samples):				36 864	122 880	245 760	552 960	983 040	2 228 224	2 228 224
Max luma sample rate (samples/sec)				552 960	3 686 400	7 372 800	16 588 800	33 177 600	66 846 720	133 693 440
Format nickname	Luma width	Luma height	Luma picture size							
SQCIF	128	96	16 384	33.7	225.0	300.0	300.0	300.0	300.0	300.0
QCIF	176	144	36 864	15.0	100.0	200.0	300.0	300.0	300.0	300.0
QVGA	320	240	81 920	-	45.0	90.0	202.5	300.0	300.0	300.0
525 SIF	352	240	98 304	-	37.5	75.0	168.7	300.0	300.0	300.0
CIF	352	288	122 880	-	30.0	60.0	135.0	270.0	300.0	300.0
525 HHR	352	480	196 608	-	-	37.5	84.3	168.7	300.0	300.0
625 HHR	352	576	221 184	-	-	33.3	75.0	150.0	300.0	300.0
Q720p	640	360	245 760	-	-	30.0	67.5	135.0	272.0	300.0
VGA	640	480	327 680	-	-	-	50.6	101.2	204.0	300.0
525 4SIF	704	480	360 448	-	-	-	46.0	92.0	185.4	300.0
525 SD	720	480	393 216	-	-	-	42.1	84.3	170.0	300.0
4CIF	704	576	405 504	-	-	-	40.9	81.8	164.8	300.0
625 SD	720	576	442 368	-	-	-	37.5	75.0	151.1	300.0
480p (16:9)	864	480	458 752	-	-	-	36.1	72.3	145.7	291.4
SVGA	800	600	532 480	-	-	-	31.1	62.3	125.5	251.0
QHD	960	540	552 960	-	-	-	30.0	60.0	120.8	241.7
XGA	1 024	768	786 432	-	-	-	-	42.1	85.0	170.0
720p HD	1 280	720	983 040	-	-	-	-	33.7	68.0	136.0
4VGA	1 280	960	1 228 800	-	-	-	-	-	54.4	108.8
SXGA	1 280	1 024	1 310 720	-	-	-	-	-	51.0	102.0
525 16SIF	1 408	960	1 351 680	-	-	-	-	-	49.4	98.9
16CIF	1 408	1 152	1 622 016	-	-	-	-	-	41.2	82.4
4SVGA	1 600	1 200	1 945 600	-	-	-	-	-	34.3	68.7
1080 HD	1 920	1 080	2 088 960	-	-	-	-	-	32.0	64.0
2Kx1K	2 048	1 024	2 097 152	-	-	-	-	-	31.8	63.7
2Kx1080	2 048	1 080	2 228 224	-	-	-	-	-	30.0	60.0
4XGA	2 048	1 536	3 145 728	-	-	-	-	-	-	-
16VGA	2 560	1 920	4 915 200	-	-	-	-	-	-	-
3616x1536 (2.35:1)	3 616	1 536	5 603 328	-	-	-	-	-	-	-
3672x1536 (2.39:1)	3 680	1 536	5 701 632	-	-	-	-	-	-	-
3840x2160 (4*HD)	3 840	2 160	8 355 840	-	-	-	-	-	-	-
4Kx2K	4 096	2 048	8 388 608	-	-	-	-	-	-	-
4096x2160	4 096	2 160	8 912 896	-	-	-	-	-	-	-
4096x2304 (16:9)	4 096	2 304	9 437 184	-	-	-	-	-	-	-
7680x4320	7 680	4 320	33 423 360	-	-	-	-	-	-	-
8192x4096	8 192	4 096	33 554 432	-	-	-	-	-	-	-
8192x4320	8 192	4 320	35 651 584	-	-	-	-	-	-	-

**Table A.12 – Maximum picture rates (pictures per second) at level 5 to 6.2 for some example picture sizes
when MinCbSizeY is equal to 64**

Level:				5	5.1	5.2	6	6.1	6.2
Max luma picture size (samples):				8 912 896	8 912 896	8 912 896	35 651 584	35 651 584	35 651 584
Max luma sample rate (samples/sec)				267 386 880	534 773 760	1 069 547 520	1 069 547 520	2 139 095 040	4 278 190 080
Format nickname	Luma width	Luma height	Luma picture size						
SQCIF	128	96	16 384	300.0	300.0	300.0	300.0	300.0	300.0
QCIF	176	144	36 864	300.0	300.0	300.0	300.0	300.0	300.0
QVGA	320	240	81 920	300.0	300.0	300.0	300.0	300.0	300.0
525 SIF	352	240	98 304	300.0	300.0	300.0	300.0	300.0	300.0
CIF	352	288	122 880	300.0	300.0	300.0	300.0	300.0	300.0
525 HHR	352	480	196 608	300.0	300.0	300.0	300.0	300.0	300.0
625 HHR	352	576	221 184	300.0	300.0	300.0	300.0	300.0	300.0
Q720p	640	360	245 760	300.0	300.0	300.0	300.0	300.0	300.0
VGA	640	480	327 680	300.0	300.0	300.0	300.0	300.0	300.0
525 4SIF	704	480	360 448	300.0	300.0	300.0	300.0	300.0	300.0
525 SD	720	480	393 216	300.0	300.0	300.0	300.0	300.0	300.0
4CIF	704	576	405 504	300.0	300.0	300.0	300.0	300.0	300.0
625 SD	720	576	442 368	300.0	300.0	300.0	300.0	300.0	300.0
480p (16:9)	864	480	458 752	300.0	300.0	300.0	300.0	300.0	300.0
SVGA	800	600	532 480	300.0	300.0	300.0	300.0	300.0	300.0
QHD	960	540	552 960	300.0	300.0	300.0	300.0	300.0	300.0
XGA	1 024	768	786 432	300.0	300.0	300.0	300.0	300.0	300.0
720p HD	1 280	720	983 040	272.0	300.0	300.0	300.0	300.0	300.0
4VGA	1 280	960	1 228 800	217.6	300.0	300.0	300.0	300.0	300.0
SXGA	1 280	1 024	1 310 720	204.0	300.0	300.0	300.0	300.0	300.0
525 16SIF	1 408	960	1 351 680	197.8	300.0	300.0	300.0	300.0	300.0
16CIF	1 408	1 152	1 622 016	164.8	300.0	300.0	300.0	300.0	300.0
4SVGA	1 600	1 200	1 945 600	137.4	274.8	300.0	300.0	300.0	300.0
1080 HD	1 920	1 080	2 088 960	128.0	256.0	300.0	300.0	300.0	300.0
2Kx1K	2 048	1 024	2 097 152	127.5	255.0	300.0	300.0	300.0	300.0
2Kx1080	2 048	1 080	2 228 224	120.0	240.0	300.0	300.0	300.0	300.0
4XGA	2 048	1 536	3 145 728	85.0	170.0	300.0	300.0	300.0	300.0
16VGA	2 560	1 920	4 915 200	54.4	108.8	217.6	217.6	300.0	300.0
3616x1536 (2.35:1)	3 616	1 536	5 603 328	47.7	95.4	190.8	190.8	300.0	300.0
3672x1536 (2.39:1)	3 680	1 536	5 701 632	46.8	93.7	187.5	187.5	300.0	300.0
3840x2160 (4*HD)	3 840	2 160	8 355 840	32.0	64.0	128.0	128.0	256.0	300.0
4Kx2K	4 096	2 048	8 388 608	31.8	63.7	127.5	127.5	255.0	300.0
4096x2160	4 096	2 160	8 912 896	30.0	60.0	120.0	120.0	240.0	300.0
4096x2304 (16:9)	4 096	2 304	9 437 184	-	-	-	113.3	226.6	300.0
4096x3072	4 096	3 072	12 582 912	-	-	-	85.0	170.0	300.0
7680x4320	7 680	4 320	33 423 360	-	-	-	32.0	64.0	128.0
8192x4096	8 192	4 096	33 554 432	-	-	-	31.8	63.7	127.5
8192x4320	8 192	4 320	35 651 584	-	-	-	30.0	60.0	120.0

The following should be noted in regard to the examples shown in Tables A.11 and A.12:

- This is a variable-picture-size Specification. The specific listed picture sizes are illustrative examples only.
- The example luma picture sizes were computed by rounding up the luma width and luma height to multiples of 64 before computing the product of these quantities, to reflect the potential use of MinCbSizeY equal to 64 for these picture sizes, as pic_width_in_luma_samples and pic_height_in_luma_samples are each required to be a multiple of MinCbSizeY. For some illustrated values of luma width and luma height, a somewhat higher number of pictures per second can be supported when MinCbSizeY is less than 64.

- In cases where the maximum picture rate value is not an integer multiple of 0.1 pictures per second, the given maximum picture rate values have been rounded down to the largest integer multiple of 0.1 frames per second that does not exceed the exact value. For example, for level 3.1, the maximum picture rate for 720p HD has been rounded down to 33.7 from an exact value of 33.75.
- As used in the examples, "525" refers to typical use for environments using 525 analogue scan lines (of which approximately 480 lines contain the visible picture region) and "625" refers to environments using 625 analogue scan lines (of which approximately 576 lines contain the visible picture region).
- XGA is also known as (aka) XVGA, 4SVGA aka UXGA, 16XGA aka 4Kx3K, CIF aka 625 SIF, 625 HHR aka 2CIF aka half 625 D-1, aka half 625 ITU-R BT.601, 525 SD aka 525 D-1 aka 525 ITU-R BT.601, 625 SD aka 625 D-1 aka 625 ITU-R BT.601.

Annex B

Byte stream format

(This annex forms an integral part of this Recommendation | International Standard.)

B.1 General

This annex specifies syntax and semantics of a byte stream format specified for use by applications that deliver some or all of the NAL unit stream as an ordered stream of bytes or bits within which the locations of NAL unit boundaries need to be identifiable from patterns in the data, such as Rec. ITU-T H.222.0 | ISO/IEC 13818-1 systems or Recommendation ITU-T H.320 systems. For bit-oriented delivery, the bit order for the byte stream format is specified to start with the MSB of the first byte, proceed to the LSB of the first byte, followed by the MSB of the second byte, etc.

The byte stream format consists of a sequence of byte stream NAL unit syntax structures. Each byte stream NAL unit syntax structure contains one start code prefix followed by one `nal_unit(NumBytesInNalUnit)` syntax structure. It may (and under some circumstances, it shall) also contain an additional `zero_byte` syntax element. It may also contain one or more additional trailing `zero_8bits` syntax elements. When it is the first byte stream NAL unit in the bitstream, it may also contain one or more additional leading `zero_8bits` syntax elements.

B.2 Byte stream NAL unit syntax and semantics

B.2.1 Byte stream NAL unit syntax

<code>byte_stream_nal_unit(NumBytesInNalUnit) {</code>	Descriptor
<code>while(next_bits(24) != 0x000001 && next_bits(32) != 0x00000001)</code>	
<code> leading_zero_8bits /* equal to 0x00 */</code>	f(8)
<code>if(next_bits(24) != 0x000001)</code>	
<code> zero_byte /* equal to 0x00 */</code>	f(8)
<code> start_code_prefix_one_3bytes /* equal to 0x000001 */</code>	f(24)
<code> nal_unit(NumBytesInNalUnit)</code>	
<code> while(more_data_in_byte_stream() && next_bits(24) != 0x000001 && next_bits(32) != 0x00000001)</code>	
<code> trailing_zero_8bits /* equal to 0x00 */</code>	f(8)
<code>}</code>	

B.2.2 Byte stream NAL unit semantics

The order of byte stream NAL units in the byte stream shall follow the decoding order of the NAL units contained in the byte stream NAL units (see clause 7.4.2.4). The content of each byte stream NAL unit is associated with the same access unit as the NAL unit contained in the byte stream NAL unit (see clause 7.4.2.4.4).

leading_zero_8bits is a byte equal to 0x00.

NOTE – The `leading_zero_8bits` syntax element can only be present in the first byte stream NAL unit of the bitstream, because (as shown in the syntax diagram of clause B.2.1) any bytes equal to 0x00 that follow a NAL unit syntax structure and precede the four-byte sequence 0x00000001 (which is to be interpreted as a `zero_byte` followed by a `start_code_prefix_one_3bytes`) will be considered to be `trailing_zero_8bits` syntax elements that are part of the preceding byte stream NAL unit.

zero_byte is a single byte equal to 0x00.

When one or more of the following conditions are true, the `zero_byte` syntax element shall be present:

- The `nal_unit_type` within the `nal_unit()` syntax structure is equal to `VPS_NUT`, `SPS_NUT` or `PPS_NUT`.
- The byte stream NAL unit syntax structure contains the first NAL unit of an access unit in decoding order, as specified in clause 7.4.2.4.4.

start_code_prefix_one_3bytes is a fixed-value sequence of 3 bytes equal to 0x000001. This syntax element is called a start code prefix.

trailing_zero_8bits is a byte equal to 0x00.

B.3 Byte stream NAL unit decoding process

Input to this process consists of an ordered stream of bytes consisting of a sequence of byte stream NAL unit syntax structures.

Output of this process consists of a sequence of NAL unit syntax structures.

At the beginning of the decoding process, the decoder initializes its current position in the byte stream to the beginning of the byte stream. It then extracts and discards each `leading_zero_8bits` syntax element (when present), moving the current position in the byte stream forward one byte at a time, until the current position in the byte stream is such that the next four bytes in the bitstream form the four-byte sequence `0x00000001`.

The decoder then performs the following step-wise process repeatedly to extract and decode each NAL unit syntax structure in the byte stream until the end of the byte stream has been encountered (as determined by unspecified means) and the last NAL unit in the byte stream has been decoded:

1. When the next four bytes in the bitstream form the four-byte sequence `0x00000001`, the next byte in the byte stream (which is a `zero_byte` syntax element) is extracted and discarded and the current position in the byte stream is set equal to the position of the byte following this discarded byte.
2. The next three-byte sequence in the byte stream (which is a `start_code_prefix_one_3bytes`) is extracted and discarded and the current position in the byte stream is set equal to the position of the byte following this three-byte sequence.
3. `NumBytesInNalUnit` is set equal to the number of bytes starting with the byte at the current position in the byte stream up to and including the last byte that precedes the location of one or more of the following conditions:
 - A subsequent byte-aligned three-byte sequence equal to `0x0000000`,
 - A subsequent byte-aligned three-byte sequence equal to `0x0000001`,
 - The end of the byte stream, as determined by unspecified means.
4. `NumBytesInNalUnit` bytes are removed from the bitstream and the current position in the byte stream is advanced by `NumBytesInNalUnit` bytes. This sequence of bytes is `nal_unit(NumBytesInNalUnit)` and is decoded using the NAL unit decoding process.
5. When the current position in the byte stream is not at the end of the byte stream (as determined by unspecified means) and the next bytes in the byte stream do not start with a three-byte sequence equal to `0x0000001` and the next bytes in the byte stream do not start with a four byte sequence equal to `0x00000001`, the decoder extracts and discards each `trailing_zero_8bits` syntax element, moving the current position in the byte stream forward one byte at a time, until the current position in the byte stream is such that the next bytes in the byte stream form the four-byte sequence `0x00000001` or the end of the byte stream has been encountered (as determined by unspecified means).

B.4 Decoder byte-alignment recovery (informative)

This clause does not form an integral part of this Specification.

Many applications provide data to a decoder in a manner that is inherently byte aligned, and thus have no need for the bit-oriented byte alignment detection procedure described in this clause.

A decoder is said to have byte alignment with a bitstream when the decoder has determined whether or not the positions of data in the bitstream are byte-aligned. When a decoder does not have byte alignment with the bitstream, the decoder may examine the incoming bitstream for the binary pattern '00000000 00000000 00000000 00000001' (31 consecutive bits equal to 0 followed by a bit equal to 1). The bit immediately following this pattern is the first bit of an aligned byte following a start code prefix. Upon detecting this pattern, the decoder will be byte-aligned with the bitstream and positioned at the start of a NAL unit in the bitstream.

Once byte aligned with the bitstream, the decoder can examine the incoming bitstream data for subsequent three-byte sequences `0x0000001` and `0x0000003`.

When the three-byte sequence `0x0000001` is detected, this is a start code prefix.

When the three-byte sequence `0x0000003` is detected, the third byte (`0x03`) is an `emulation_prevention_three_byte` to be discarded as specified in clause 7.4.2.

When an error in the bitstream syntax is detected (e.g., a non-zero value of the `forbidden_zero_bit` or one of the three-byte or four-byte sequences that are prohibited in clause 7.4.2), the decoder may consider the detected condition as an indication that byte alignment may have been lost and may discard all bitstream data until the detection of byte alignment at a later position in the bitstream as described above in this clause.

Annex C

Hypothetical reference decoder

(This annex forms an integral part of this Recommendation | International Standard.)

C.1 General

This annex specifies the hypothetical reference decoder (HRD) and its use to check bitstream and decoder conformance.

Two types of bitstreams or bitstream subsets are subject to HRD conformance checking for this Specification. The first type, called a Type I bitstream, is a NAL unit stream containing only the VCL NAL units and NAL units with `nal_unit_type` equal to `FD_NUT` (filler data NAL units) for all access units in the bitstream. The second type, called a Type II bitstream, contains, in addition to the VCL NAL units and filler data NAL units for all access units in the bitstream, at least one of the following:

- additional non-VCL NAL units other than filler data NAL units,
- all `leading_zero_8bits`, `zero_byte`, `start_code_prefix_one_3bytes` and `trailing_zero_8bits` syntax elements that form a byte stream from the NAL unit stream (as specified in Annex B).

NOTE 1 – Decoders conforming to profiles specified in Annex A do not use NAL units with `nuh_layer_id` greater than 0 (e.g., access unit delimiter NAL units with `nuh_layer_id` greater than 0) for access unit boundary detection, except for identification of whether a NAL unit is a VCL or non-VCL NAL unit. Consequently, hypothetical reference decoder (HRD) parameters carried in non-scalable-nested buffering period, picture timing and decoding unit information SEI messages apply to access units that are identified based on such access unit boundary detection.

Figure C.1 shows the types of bitstream conformance points checked by the HRD.

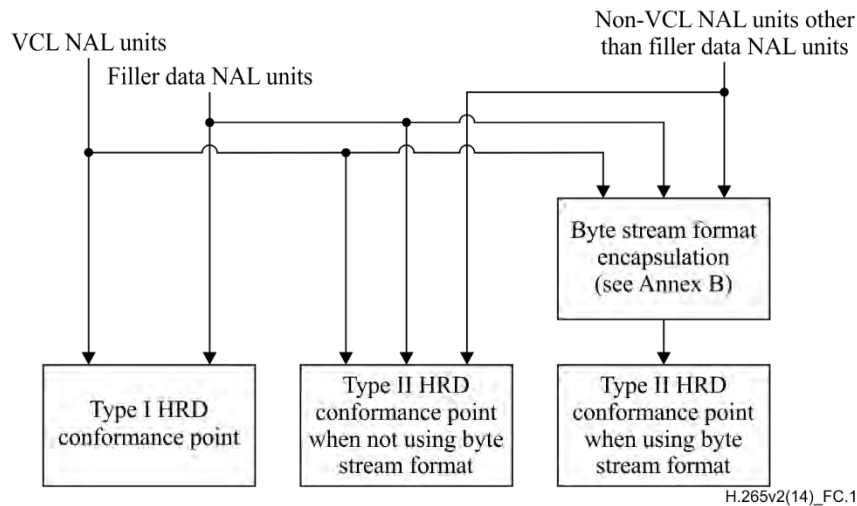


Figure C.1 – Structure of byte streams and NAL unit streams for HRD conformance checks

The syntax elements of non-VCL NAL units (or their default values for some of the syntax elements), required for the HRD, are specified in the semantic clauses of clause 7, Annexes D and E.

Two types of HRD parameter sets (NAL HRD parameters and VCL HRD parameters) are used. The HRD parameter sets are signalled through the `hrd_parameters()` syntax structure, which may be part of the SPS syntax structure or the VPS syntax structure.

Two sets of bitstream conformance tests are needed for checking the conformance of a bitstream, which is referred to as the entire bitstream, denoted as `entireBitstream`. The first set of bitstream conformance tests are for testing the conformance of the entire bitstream and its temporal subsets, regardless of whether there is a layer set specified by the active VPS that contains all the `nuh_layer_id` values of VCL NAL units present in the entire bitstream. The second set of bitstream conformance tests are for testing the conformance of the layer sets specified by the active VPS and their temporal subsets. For all these tests, only the base layer pictures (i.e., pictures with `nuh_layer_id` equal to 0) are decoded and other pictures are ignored by the decoder when the decoding process is invoked.

For each test, the following ordered steps apply in the order listed, followed by the processes described after these steps in this clause:

1. An operation point under test, denoted as TargetOp, is selected by selecting a layer identifier list OpLayerIdList and a target highest TemporalId value OpTid. The layer identifier list OpLayerIdList of TargetOp either consists of all the nuh_layer_id values of the VCL NAL units present in entireBitstream or consists of all the nuh_layer_id values of a layer set specified by the active VPS. The value of OpTid is in the range of 0 to sps_max_sub_layers_minus1, inclusive. The values of OpLayerIdList and OpTid are such that the sub-bitstream BitstreamToDecode that is the output by invoking the sub-bitstream extraction process as specified in clause 10 with entireBitstream, OpTid and OpLayerIdList as inputs satisfy both of the following conditions:
 - There is at least one VCL NAL unit in BitstreamToDecode with nuh_layer_id equal to each of the nuh_layer_id values in OpLayerIdList.
 - There is at least one VCL NAL unit with TemporalId equal to OpTid in BitstreamToDecode.
2. TargetDecLayerIdList is set equal to OpLayerIdList of TargetOp and HighestTid is set equal to OpTid of TargetOp.
3. The hrd_parameters() syntax structure and the sub_layer_hrd_parameters() syntax structure applicable to TargetOp are selected. If TargetDecLayerIdList contains all nuh_layer_id values present in entireBitstream, the hrd_parameters() syntax structure in the active SPS (or provided through an external means not specified in this Specification) is selected. Otherwise, the hrd_parameters() syntax structure in the active VPS (or provided through some external means not specified in this Specification) that applies to TargetOp is selected. Within the selected hrd_parameters() syntax structure, if BitstreamToDecode is a Type I bitstream, the sub_layer_hrd_parameters(HighestTid) syntax structure that immediately follows the condition "if(vcl_hrd_parameters_present_flag)" is selected and the variable NalHrdModeFlag is set equal to 0; otherwise (BitstreamToDecode is a Type II bitstream), the sub_layer_hrd_parameters(HighestTid) syntax structure that immediately follows either the condition "if(vcl_hrd_parameters_present_flag)" (in this case the variable NalHrdModeFlag is set equal to 0) or the condition "if(nal_hrd_parameters_present_flag)" (in this case the variable NalHrdModeFlag is set equal to 1) is selected. When BitstreamToDecode is a Type II bitstream and NalHrdModeFlag is equal to 0, all non-VCL NAL units except filler data NAL units, and all leading_zero_8bits, zero_byte, start_code_prefix_one_3bytes and trailing_zero_8bits syntax elements that form a byte stream from the NAL unit stream (as specified in Annex B), when present, are discarded from BitstreamToDecode and the remaining bitstream is assigned to BitstreamToDecode.
4. An access unit associated with a buffering period SEI message (present in BitstreamToDecode or available through external means not specified in this Specification) applicable to TargetOp is selected as the HRD initialization point and referred to as access unit 0. If TargetDecLayerIdList contains all nuh_layer_id values present in entireBitstream, the associated buffering period SEI message shall be either a non-scalable-nested SEI message or provided by external means. Otherwise, the associated buffering period SEI message shall be either a scalable-nested SEI message or provided by external means.
5. When sub_pic_hrd_params_present_flag in the selected hrd_parameters() syntax structure is equal to 1, the CPB is scheduled to operate either at the access unit level (in which case the variable SubPicHrdFlag is set equal to 0) or at the sub-picture level (in which case the variable SubPicHrdFlag is set equal to 1). Otherwise, SubPicHrdFlag is set equal to 0 and the CPB is scheduled to operate at the partition unit level.
6. For each access unit in BitstreamToDecode starting from access unit 0, the buffering period SEI message (present in BitstreamToDecode or available through external means not specified in this Specification) that is associated with the access unit and applies to TargetOp is selected, the picture timing SEI message (present in BitstreamToDecode or available through external means not specified in this Specification) that is associated with the access unit and applies to TargetOp is selected, and when SubPicHrdFlag is equal to 1 and sub_pic_cpb_params_in_pic_timing_sei_flag is equal to 0, the decoding unit information SEI messages (present in BitstreamToDecode or available through external means not specified in this Specification) that are associated with decoding units in the access unit and apply to TargetOp are selected. If TargetDecLayerIdList contains all nuh_layer_id values present in entireBitstream, the selected buffering period, picture timing and decoding unit information SEI messages shall be either non-scalable-nested SEI messages or provided by external means. Otherwise, the selected buffering period, picture timing and decoding unit information SEI messages shall be either scalable-nested SEI messages or provided by external means.
7. A value of SchedSelIdx is selected. The selected SchedSelIdx shall be in the range of 0 to cpb_cnt_minus1[HighestTid], inclusive, where cpb_cnt_minus1[HighestTid] is found in the hrd_parameters() syntax structure as selected above.
8. When the coded picture in access unit 0 has nal_unit_type equal to CRA_NUT or BLA_W_LP and irap_cpb_params_present_flag in the selected buffering period SEI message is equal to 1, either of the following applies for selection of the initial CPB removal delay and delay offset:
 - If NalHrdModeFlag is equal to 1, the default initial CPB removal delay and delay offset represented by nal_initial_cpb_removal_delay[SchedSelIdx] and nal_initial_cpb_removal_offset[SchedSelIdx], respectively, in the selected buffering period SEI message are selected. Otherwise, the default initial CPB

removal delay and delay offset represented by `vcl_initial_cpb_removal_delay[SchedSelIdx]` and `vcl_initial_cpb_removal_offset[SchedSelIdx]`, respectively, in the selected buffering period SEI message are selected. The variable `DefaultInitCpbParamsFlag` is set equal to 1.

- If `NalHrdModeFlag` is equal to 1, the alternative initial CPB removal delay and delay offset represented by `nal_initial_alt_cpb_removal_delay[SchedSelIdx]` and `nal_initial_alt_cpb_removal_offset[SchedSelIdx]`, respectively, in the selected buffering period SEI message are selected. Otherwise, the alternative initial CPB removal delay and delay offset represented by `vcl_initial_alt_cpb_removal_delay[SchedSelIdx]` and `vcl_initial_alt_cpb_removal_offset[SchedSelIdx]`, respectively, in the selected buffering period SEI message are selected. The variable `DefaultInitCpbParamsFlag` is set equal to 0, and the RASL access units associated with access unit 0 are discarded from `BitstreamToDecode` and the remaining bitstream is assigned to `BitstreamToDecode`.

Each conformance test consists of a combination of one option in each of the above steps. When there is more than one option for a step, for any particular conformance test only one option is chosen. All possible combinations of all the steps form the entire set of conformance tests. For each operation point under test, the number of bitstream conformance tests to be performed is equal to $n0 * n1 * (n2 * 2 + n3) * n4$, where the values of $n0$, $n1$, $n2$, $n3$ and $n4$ are specified as follows:

- $n0$ is derived as follows:
 - If `BitstreamToDecode` is a Type I bitstream, $n0$ is equal to 1.
 - Otherwise (`BitstreamToDecode` is a Type II bitstream), $n0$ is equal to 2.
- $n1$ is equal to `cpb_cnt_minus1[HighestTid] + 1`.
- $n2$ is the number of access units in `BitstreamToDecode` that each is associated with a buffering period SEI message applicable to `TargetOp` and for each of which both of the following conditions are true:
 - `nal_unit_type` is equal to `CRA_NUT` or `BLA_W_LP` for the VCL NAL units.
 - The associated buffering period SEI message applicable to `TargetOp` has `irap_cpb_params_present_flag` equal to 1.
- $n3$ is the number of access units in `BitstreamToDecode` that each is associated with a buffering period SEI message applicable to `TargetOp` and for each of which one or both of the following conditions are true:
 - `nal_unit_type` is equal to neither `CRA_NUT` nor `BLA_W_LP` for the VCL NAL units.
 - The associated buffering period SEI message applicable to `TargetOp` has `irap_cpb_params_present_flag` equal to 0.
- $n4$ is derived as follows:
 - If `sub_pic_hrd_params_present_flag` in the selected `hrd_parameters()` syntax structure is equal to 0, $n4$ is equal to 1.
 - Otherwise, $n4$ is equal to 2.

When `BitstreamToDecode` is a Type II bitstream, the following applies:

- If the `sub_layer_hrd_parameters(HighestTid)` syntax structure that immediately follows the condition "`if(vcl_hrd_parameters_present_flag)`" is selected, the test is conducted at the Type I conformance point shown in Figure C.1, and only VCL and filler data NAL units are counted for the input bit rate and CPB storage.
- Otherwise (the `sub_layer_hrd_parameters(HighestTid)` syntax structure that immediately follows the condition "`if(nal_hrd_parameters_present_flag)`" is selected), the test is conducted at the Type II conformance point shown in Figure C.1, and all bytes of the Type II bitstream, which may be a NAL unit stream or a byte stream, are counted for the input bit rate and CPB storage.

NOTE 2 – NAL HRD parameters established by a value of `SchedSelIdx` for the Type II conformance point shown in Figure C.1 are sufficient to also establish VCL HRD conformance for the Type I conformance point shown in Figure C.1 for the same values of `InitCpbRemovalDelay[SchedSelIdx]`, `BitRate[SchedSelIdx]` and `CpbSize[SchedSelIdx]` for the variable bit rate (VBR) case (`cbr_flag[SchedSelIdx]` equal to 0). This is because the data flow into the Type I conformance point is a subset of the data flow into the Type II conformance point and because, for the VBR case, the CPB is allowed to become empty and stay empty until the time a next picture is scheduled to begin to arrive. For example, when decoding a CVS conforming to one or more of the profiles specified in Annex A using the decoding process specified in clauses 2 through 10, when NAL HRD parameters are provided for the Type II conformance point that not only fall within the bounds set for NAL HRD parameters for profile conformance in item f) of clause A.4.2 but also fall within the bounds set for VCL HRD parameters for profile conformance in item e) of clause A.4.2, conformance of the VCL HRD for the Type I conformance point is also assured to fall within the bounds of item e) of clause A.4.2.

All VPSs, SPSs and PPSs referred to in the VCL NAL units and the corresponding buffering period, picture timing and decoding unit information SEI messages shall be conveyed to the HRD, in a timely manner, either in the bitstream (by non-VCL NAL units), or by other means not specified in this Specification.

In Annexes C, D and E, the specification for "presence" of non-VCL NAL units that contain VPSs, SPSs, PPSs, buffering period SEI messages, picture timing SEI messages or decoding unit information SEI messages is also satisfied when those

NAL units (or just some of them) are conveyed to decoders (or to the HRD) by other means not specified in this Specification. For the purpose of counting bits, only the appropriate bits that are actually present in the bitstream are counted.

NOTE 3 – As an example, synchronization of such a non-VCL NAL unit, conveyed by means other than presence in the bitstream, with the NAL units that are present in the bitstream, can be achieved by indicating two points in the bitstream, between which the non-VCL NAL unit would have been present in the bitstream, had the encoder decided to convey it in the bitstream.

When the content of such a non-VCL NAL unit is conveyed for the application by some means other than presence within the bitstream, the representation of the content of the non-VCL NAL unit is not required to use the same syntax as specified in this Specification.

NOTE 4 – When HRD information is contained within the bitstream, it is possible to verify the conformance of a bitstream to the requirements of this clause based solely on information contained in the bitstream. When the HRD information is not present in the bitstream, as is the case for all "stand-alone" Type I bitstreams, conformance can only be verified when the HRD data are supplied by some other means not specified in this Specification.

The HRD contains a coded picture buffer (CPB), an instantaneous decoding process, a decoded picture buffer (DPB), and output cropping as shown in Figure C.2.

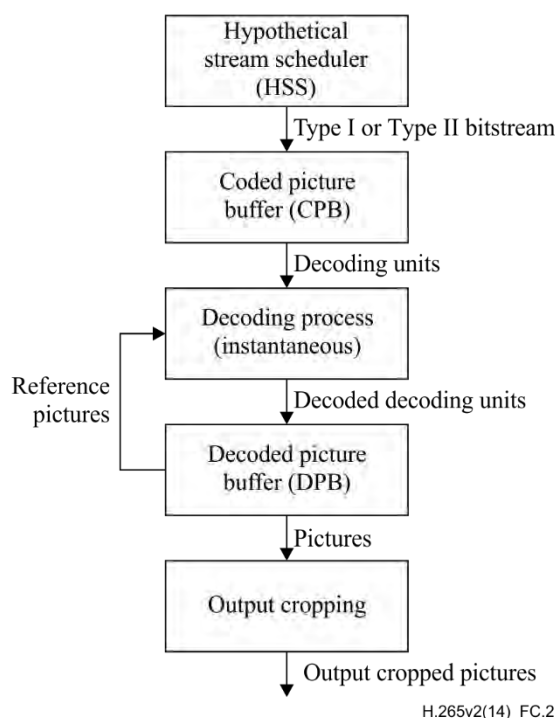


Figure C.2 – HRD buffer model

For each bitstream conformance test, the CPB size (number of bits) is $CpbSize[SchedSelIdx]$ as specified in clause E.3.3, where $SchedSelIdx$ and the HRD parameters are specified above in this clause. The DPB size (number of picture storage buffers) is $sps_max_dec_pic_buffering_minus1[HighestTid] + 1$.

If $SubPicHrdFlag$ is equal to 0, the HRD operates at access unit level and each decoding unit is an access unit. Otherwise the HRD operates at sub-picture level and each decoding unit is a subset of an access unit.

NOTE 5 – If the HRD operates at access unit level, each time when some bits are removed from the CPB, a decoding unit that is an entire access unit is removed from the CPB. Otherwise (the HRD operates at sub-picture level), each time when some bits are removed from the CPB, a decoding unit that is a subset of an access unit is removed from the CPB. Regardless of whether the HRD operates at access unit level or sub-picture level, each time when some picture is output from the DPB, an entire decoded picture is output from the DPB, though the picture output time is derived based on the differently derived CPB removal times and the differently signalled DPB output delays.

The following is specified for expressing the constraints in this annex:

- Each access unit is referred to as access unit n , where the number n identifies the particular access unit. Access unit 0 is selected per step 4 above. The value of n is incremented by 1 for each subsequent access unit in decoding order.
- Each decoding unit is referred to as decoding unit m , where the number m identifies the particular decoding unit. The first decoding unit in decoding order in access unit 0 is referred to as decoding unit 0. The value of m is incremented by 1 for each subsequent decoding unit in decoding order.

NOTE 6 – The numbering of decoding units is relative to the first decoding unit in access unit 0.

- Picture *n* refers to the coded picture or the decoded picture of access unit *n*.

The HRD operates as follows:

- The HRD is initialized at decoding unit 0, with both the CPB and the DPB being set to be empty (the DPB fullness is set equal to 0).

NOTE 7 – After initialization, the HRD is not initialized again by subsequent buffering period SEI messages.

- Data associated with decoding units that flow into the CPB according to a specified arrival schedule are delivered by the hypothetical stream scheduler (HSS).
- The data associated with each decoding unit are removed and decoded instantaneously by the instantaneous decoding process at the CPB removal time of the decoding unit.
- Each decoded picture is placed in the DPB.
- A decoded picture is removed from the DPB when it becomes no longer needed for inter prediction reference and no longer needed for output.

For each bitstream conformance test, the operation of the CPB is specified in clause C.2, the instantaneous decoder operation is specified in clauses 2 through 10, the operation of the DPB is specified in clause C.3 and the output cropping is specified in clauses C.3.3 and C.5.2.2.

HSS and HRD information concerning the number of enumerated delivery schedules and their associated bit rates and buffer sizes is specified in clauses E.2.2 and E.3.2. The HRD is initialized as specified by the buffering period SEI message specified in clauses D.2.2 and D.3.2. The removal timing of decoding units from the CPB and output timing of decoded pictures from the DPB is specified using information in picture timing SEI messages (specified in clauses D.2.3 and 0) or in decoding unit information SEI messages (specified in clauses D.2.22 and D.3.22). All timing information relating to a specific decoding unit shall arrive prior to the CPB removal time of the decoding unit.

The requirements for bitstream conformance are specified in clause C.4 and the HRD is used to check conformance of bitstreams as specified above in this clause and to check conformance of decoders as specified in clause C.5.

NOTE 8 – While conformance is guaranteed under the assumption that all picture-rates and clocks used to generate the bitstream match exactly the values signalled in the bitstream, in a real system each of these may vary from the signalled or specified value.

All the arithmetic in this annex is performed with real values, so that no rounding errors can propagate. For example, the number of bits in a CPB just prior to or after removal of a decoding unit is not necessarily an integer.

The variable *ClockTick* is derived as follows and is called a clock tick:

$$\text{ClockTick} = \text{vui_num_units_in_tick} \div \text{vui_time_scale} \quad (\text{C-1})$$

The variable *ClockSubTick* is derived as follows and is called a clock sub-tick:

$$\text{ClockSubTick} = \text{ClockTick} \div (\text{tick_divisor_minus2} + 2) \quad (\text{C-2})$$

C.2 Operation of coded picture buffer

C.2.1 General

The specifications in this clause apply independently to each set of coded picture buffer (CPB) parameters that is present and to both the Type I and Type II conformance points shown in Figure C.1 and the set of CPB parameters is selected as specified in clause C.1.

C.2.2 Timing of decoding unit arrival

If *SubPicHrdFlag* is equal to 0, the variable *subPicParamsFlag* is set equal to 0 and the process specified in the remainder of this clause is invoked with a decoding unit being considered as an access unit, for derivation of the initial and final CPB arrival times for access unit *n*.

Otherwise (*SubPicHrdFlag* is equal to 1), the process specified in the remainder of this clause is first invoked with the variable *subPicParamsFlag* set equal to 0 and a decoding unit being considered as an access unit, for derivation of the initial and final CPB arrival times for access unit *n*, and then invoked with *subPicParamsFlag* set equal to 1 and a decoding unit being considered as a subset of an access unit, for derivation of the initial and final CPB arrival times for the decoding units in access unit *n*.

The variables *InitCpbRemovalDelay[SchedSelIdx]* and *InitCpbRemovalDelayOffset[SchedSelIdx]* are derived as follows:

- If one or more of the following conditions are true, `InitCpbRemovalDelay[SchedSelIdx]` and `InitCpbRemovalDelayOffset[SchedSelIdx]` are set equal to the values of the buffering period SEI message syntax elements `nal_initial_alt_cpb_removal_delay[SchedSelIdx]` and `nal_initial_alt_cpb_removal_offset[SchedSelIdx]`, respectively, when `NalHrdModeFlag` is equal to 1 or `vcl_initial_alt_cpb_removal_delay[SchedSelIdx]` and `vcl_initial_alt_cpb_removal_offset[SchedSelIdx]`, respectively, when `NalHrdModeFlag` is equal to 0, where the buffering period SEI message syntax elements are selected as specified in clause C.1:
 - Access unit 0 is a BLA access unit for which the coded picture has `nal_unit_type` equal to `BLA_W_RADL` or `BLA_N_LP`, and the value of `irap_cpb_params_present_flag` of the buffering period SEI message is equal to 1.
 - Access unit 0 is a BLA access unit for which the coded picture has `nal_unit_type` equal to `BLA_W_LP` or is a CRA access unit, and the value of `irap_cpb_params_present_flag` of the buffering period SEI message is equal to 1 and one or more of the following conditions are true:
 - `UseAltCpbParamsFlag` for access unit 0 is equal to 1.
 - `DefaultInitCpbParamsFlag` is equal to 0.
 - The value of `subPicParamsFlag` is equal to 1.
- Otherwise, `InitCpbRemovalDelay[SchedSelIdx]` and `InitCpbRemovalDelayOffset[SchedSelIdx]` are set equal to the values of the buffering period SEI message syntax elements `nal_initial_cpb_removal_delay[SchedSelIdx]` and `nal_initial_cpb_removal_offset[SchedSelIdx]`, respectively, when `NalHrdModeFlag` is equal to 1, or `vcl_initial_cpb_removal_delay[SchedSelIdx]` and `vcl_initial_cpb_removal_offset[SchedSelIdx]`, respectively, when `NalHrdModeFlag` is equal to 0, where the buffering period SEI message syntax elements are selected as specified in clause C.1.

The time at which the first bit of decoding unit `m` begins to enter the CPB is referred to as the initial arrival time `initArrivalTime[m]`.

The initial arrival time of decoding unit `m` is derived as follows:

- If the decoding unit is decoding unit 0 (i.e., when `m` is equal to 0), `initArrivalTime[0]` is set equal to 0.
- Otherwise (the decoding unit is decoding unit `m` with `m > 0`), the following applies:
 - If `cbr_flag[SchedSelIdx]` is equal to 1, the initial arrival time for decoding unit `m` is equal to the final arrival time (which is derived below) of decoding unit `m – 1`, i.e.,

$$\begin{aligned}
 &\text{if(!subPicParamsFlag)} \\
 &\quad \text{initArrivalTime[m]} = \text{AuFinalArrivalTime[m – 1]} \\
 &\text{else} \\
 &\quad \text{initArrivalTime[m]} = \text{DuFinalArrivalTime[m – 1]}
 \end{aligned}
 \tag{C-3}$$

- Otherwise (`cbr_flag[SchedSelIdx]` is equal to 0), the initial arrival time for decoding unit `m` is derived as follows:

$$\begin{aligned}
 &\text{if(!subPicParamsFlag)} \\
 &\quad \text{initArrivalTime[m]} = \text{Max(AuFinalArrivalTime[m – 1], initArrivalEarliestTime[m])} \\
 &\text{(C-4)} \\
 &\text{else} \\
 &\quad \text{initArrivalTime[m]} = \text{Max(DuFinalArrivalTime[m – 1], initArrivalEarliestTime[m])}
 \end{aligned}$$

where `initArrivalEarliestTime[m]` is derived as follows:

- The variable `tmpNominalRemovalTime` is derived as follows:

$$\begin{aligned}
 &\text{if(!subPicParamsFlag)} \\
 &\quad \text{tmpNominalRemovalTime} = \text{AuNominalRemovalTime[m]} \\
 &\text{else} \\
 &\quad \text{tmpNominalRemovalTime} = \text{DuNominalRemovalTime[m]}
 \end{aligned}
 \tag{C-5}$$

where `AuNominalRemovalTime[m]` and `DuNominalRemovalTime[m]` are the nominal CPB removal time of access unit `m` and decoding unit `m`, respectively, as specified in clause C.2.3.

- If decoding unit `m` is not the first decoding unit of a subsequent buffering period, `initArrivalEarliestTime[m]` is derived as follows:

$$\begin{aligned} \text{initArrivalEarliestTime}[m] = & \text{tmpNominalRemovalTime} - \\ & (\text{InitCpbRemovalDelay}[\text{SchedSelIdx}] \\ & + \text{InitCpbRemovalDelayOffset}[\text{SchedSelIdx}]) \div 90\,000 \end{aligned} \quad (\text{C-6})$$

- Otherwise (decoding unit m is the first decoding unit of a subsequent buffering period), $\text{initArrivalEarliestTime}[m]$ is derived as follows:

$$\begin{aligned} \text{initArrivalEarliestTime}[m] = & \text{tmpNominalRemovalTime} - \\ & (\text{InitCpbRemovalDelay}[\text{SchedSelIdx}] \div 90\,000) \end{aligned} \quad (\text{C-7})$$

The final arrival time for decoding unit m is derived as follows:

$$\begin{aligned} & \text{if(!subPicParamsFlag)} \\ & \quad \text{AuFinalArrivalTime}[m] = \text{initArrivalTime}[m] + \text{sizeInbits}[m] \div \text{BitRate}[\text{SchedSelIdx}] \quad (\text{C-8}) \\ & \text{else} \\ & \quad \text{DuFinalArrivalTime}[m] = \text{initArrivalTime}[m] + \text{sizeInbits}[m] \div \text{BitRate}[\text{SchedSelIdx}] \end{aligned}$$

where $\text{sizeInbits}[m]$ is the size in bits of decoding unit m , counting the bits of the VCL NAL units and the filler data NAL units for the Type I conformance point or all bits of the Type II bitstream for the Type II conformance point, where the Type I and Type II conformance points are as shown in Figure C.1.

The values of SchedSelIdx , $\text{BitRate}[\text{SchedSelIdx}]$ and $\text{CpbSize}[\text{SchedSelIdx}]$ are constrained as follows:

- If the content of the selected $\text{hrd_parameters}()$ syntax structures for the access unit containing decoding unit m and the previous access unit differ, the HSS selects a value SchedSelIdx1 of SchedSelIdx from among the values of SchedSelIdx provided in the selected $\text{hrd_parameters}()$ syntax structures for the access unit containing decoding unit m that results in a $\text{BitRate}[\text{SchedSelIdx1}]$ or $\text{CpbSize}[\text{SchedSelIdx1}]$ for the access unit containing decoding unit m . The value of $\text{BitRate}[\text{SchedSelIdx1}]$ or $\text{CpbSize}[\text{SchedSelIdx1}]$ may differ from the value of $\text{BitRate}[\text{SchedSelIdx0}]$ or $\text{CpbSize}[\text{SchedSelIdx0}]$ for the value SchedSelIdx0 of SchedSelIdx that was in use for the previous access unit.
- Otherwise, the HSS continues to operate with the previous values of SchedSelIdx , $\text{BitRate}[\text{SchedSelIdx}]$ and $\text{CpbSize}[\text{SchedSelIdx}]$.

When the HSS selects values of $\text{BitRate}[\text{SchedSelIdx}]$ or $\text{CpbSize}[\text{SchedSelIdx}]$ that differ from those of the previous access unit, the following applies:

- The variable $\text{BitRate}[\text{SchedSelIdx}]$ comes into effect at the initial CPB arrival time of the current access unit.
- The variable $\text{CpbSize}[\text{SchedSelIdx}]$ comes into effect as follows:
 - If the new value of $\text{CpbSize}[\text{SchedSelIdx}]$ is greater than the old CPB size, it comes into effect at the initial CPB arrival time of the current access unit.
 - Otherwise, the new value of $\text{CpbSize}[\text{SchedSelIdx}]$ comes into effect at the CPB removal time of the current access unit.

C.2.3 Timing of decoding unit removal and decoding of decoding unit

The variables $\text{InitCpbRemovalDelay}[\text{SchedSelIdx}]$ and $\text{InitCpbRemovalDelayOffset}[\text{SchedSelIdx}]$ are updated, and the variables CpbDelayOffset and DpbDelayOffset are derived, as follows:

- If one or more of the following conditions are true, CpbDelayOffset is set equal to the value of the buffering period SEI message syntax element cpb_delay_offset , DpbDelayOffset is set equal to the value of the buffering period SEI message syntax element dpb_delay_offset , and $\text{InitCpbRemovalDelay}[\text{SchedSelIdx}]$ and $\text{InitCpbRemovalDelayOffset}[\text{SchedSelIdx}]$ are set equal to the values of the buffering period SEI message syntax elements $\text{nal_initial_alt_cpb_removal_delay}[\text{SchedSelIdx}]$ and $\text{nal_initial_alt_cpb_removal_offset}[\text{SchedSelIdx}]$, respectively, when NalHrdModeFlag is equal to 1, or $\text{vcl_initial_alt_cpb_removal_delay}[\text{SchedSelIdx}]$ and $\text{vcl_initial_alt_cpb_removal_offset}[\text{SchedSelIdx}]$, respectively, when NalHrdModeFlag is equal to 0, where the buffering period SEI message containing the syntax elements is selected as specified in clause C.1:
 - Access unit 0 is a BLA access unit for which the coded picture has nal_unit_type equal to BLA_W_RADL or BLA_N_LP and the value of $\text{irap_cpb_params_present_flag}$ of the buffering period SEI message is equal to 1.
 - Access unit 0 is a BLA access unit for which the coded picture has nal_unit_type equal to BLA_W_LP or is a CRA access unit and the value of $\text{irap_cpb_params_present_flag}$ of the buffering period SEI message is equal to 1 and one or more of the following conditions are true:

- UseAltCpbParamsFlag for access unit 0 is equal to 1.
- DefaultInitCpbParamsFlag is equal to 0.
- Otherwise, InitCpbRemovalDelay[SchedSelIdx] and InitCpbRemovalDelayOffset[SchedSelIdx] are set equal to the values of the buffering period SEI message syntax elements nal_initial_cpb_removal_delay[SchedSelIdx] and nal_initial_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 1, or vcl_initial_cpb_removal_delay[SchedSelIdx] and vcl_initial_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 0, where the buffering period SEI message containing the syntax elements is selected as specified in clause C.1, CpbDelayOffset and DpbDelayOffset are both set equal to 0.

The nominal removal time of the access unit n from the CPB is specified as follows:

- If access unit n is the access unit with n equal to 0 (the access unit that initializes the HRD), the nominal removal time of the access unit from the CPB is specified by:

$$\text{AuNominalRemovalTime}[0] = \text{InitCpbRemovalDelay}[\text{SchedSelIdx}] \div 90\,000 \quad (\text{C-9})$$

- Otherwise, the following applies:
 - When access unit n is the first access unit of a buffering period that does not initialize the HRD, the following applies:

The nominal removal time of the access unit n from the CPB is specified by:

$$\begin{aligned}
 & \text{if(!concatenationFlag) } \{ \\
 & \quad \text{baseTime} = \text{AuNominalRemovalTime}[\text{firstPicInPrevBuffPeriod}] \\
 & \quad \text{tmpCpbRemovalDelay} = \text{AuCpbRemovalDelayVal} \\
 & \quad \text{tmpCpbDelayOffset} = \text{CpbDelayOffset} \\
 & \} \text{ else } \{ \\
 & \quad \text{baseTime1} = \text{AuNominalRemovalTime}[\text{prevNonDiscardablePic}] \\
 & \quad \text{tmpCpbRemovalDelay1} = (\text{auCpbRemovalDelayDeltaMinus1} + 1) \\
 & \quad \text{baseTime2} = \text{AuNominalRemovalTime}[n - 1] \\
 & \quad \text{tmpCpbRemovalDelay2} = \text{Ceil}((\text{InitCpbRemovalDelay}[\text{SchedSelIdx}] \div 90\,000 + \\
 & \quad \quad \text{AuFinalArrivalTime}[n - 1] - \text{AuNominalRemovalTime}[n - 1]) \div \text{ClockTick})) \\
 & \quad \text{if(baseTime1 + ClockTick * tmpCpbRemovalDelay1 < } \\
 & \quad \quad \text{baseTime2 + ClockTick * tmpCpbRemovalDelay2) } \{ \\
 & \quad \quad \text{baseTime} = \text{baseTime2} \\
 & \quad \quad \text{tmpCpbRemovalDelay} = \text{tmpCpbRemovalDelay2} \\
 & \quad \} \text{ else } \{ \\
 & \quad \quad \text{baseTime} = \text{baseTime1} \\
 & \quad \quad \text{tmpCpbRemovalDelay} = \text{tmpCpbRemovalDelay1} \\
 & \quad \} \\
 & \quad \text{tmpCpbDelayOffset} = 0 \\
 & \} \\
 & \text{AuNominalRemovalTime}[n] = \text{baseTime} + \\
 & \quad \text{ClockTick} * (\text{tmpCpbRemovalDelay} - \text{tmpCpbDelayOffset})
 \end{aligned} \quad (\text{C-10})$$

where AuNominalRemovalTime[firstPicInPrevBuffPeriod] is the nominal removal time of the first access unit of the previous buffering period, AuNominalRemovalTime[prevNonDiscardablePic] is the nominal removal time of the preceding picture in decoding order with TemporalId equal to 0 that is not a RASL, RADL or SLNR picture, AuCpbRemovalDelayVal is the value of AuCpbRemovalDelayVal derived according to au_cpb_removal_delay_minus1 in the picture timing SEI message, selected as specified in clause C.1, associated with access unit n and concatenationFlag and auCpbRemovalDelayDeltaMinus1 are the values of the syntax elements concatenation_flag and au_cpb_removal_delay_delta_minus1, respectively, in the buffering period SEI message, selected as specified in clause C.1, associated with access unit n.

After the derivation of the nominal CPB removal time and before the derivation of the DPB output time of access unit n, the values of CpbDelayOffset and DpbDelayOffset are updated as follows:

- If one or more of the following conditions are true, CpbDelayOffset is set equal to the value of the buffering period SEI message syntax element cpb_delay_offset, and DpbDelayOffset is set equal to the value of the buffering period SEI message syntax element dpb_delay_offset, where the buffering period SEI message containing the syntax elements is selected as specified in clause C.1:

- Access unit n is a BLA access unit for which the coded picture has `nal_unit_type` equal to `BLA_W_RADL` or `BLA_N_LP` and the value of `irap_cpb_params_present_flag` of the buffering period SEI message is equal to 1.
- Access unit n is a BLA access unit for which the coded picture has `nal_unit_type` equal to `BLA_W_LP` or is a CRA access unit and the value of `irap_cpb_params_present_flag` of the buffering period SEI message is equal to 1 and `UseAltCpbParamsFlag` for access unit n is equal to 1.
- Otherwise, `CpbDelayOffset` and `DpbDelayOffset` are both set equal to 0.
- When access unit n is not the first access unit of a buffering period, the nominal removal time of the access unit n from the CPB is specified by:

$$\text{AuNominalRemovalTime}[n] = \text{AuNominalRemovalTime}[\text{firstPicInCurrBuffPeriod}] + \text{ClockTick} * (\text{AuCpbRemovalDelayVal} - \text{CpbDelayOffset}) \quad (\text{C-11})$$

where $\text{AuNominalRemovalTime}[\text{firstPicInCurrBuffPeriod}]$ is the nominal removal time of the first access unit of the current buffering period and $\text{AuCpbRemovalDelayVal}$ is the value of `AuCpbRemovalDelayVal` derived according to `au_cpb_removal_delay_minus1` in the picture timing SEI message, selected as specified in clause C.1, associated with access unit n .

When `SubPicHrdFlag` is equal to 1, the following applies:

- The variable `duCpbRemovalDelayInc` is derived as follows:
 - If `sub_pic_cpb_params_in_pic_timing_sei_flag` is equal to 0, `duCpbRemovalDelayInc` is set equal to the value of `du_spt_cpb_removal_delay_increment` in the decoding unit information SEI message, selected as specified in clause C.1, associated with decoding unit m .
 - Otherwise, if `du_common_cpb_removal_delay_flag` is equal to 0, `duCpbRemovalDelayInc` is set equal to the value of `du_cpb_removal_delay_increment_minus1[i] + 1` for decoding unit m in the picture timing SEI message, selected as specified in clause C.1, associated with access unit n , where the value of i is 0 for the first `num_nalus_in_du_minus1[0] + 1` consecutive NAL units in the access unit that contains decoding unit m , 1 for the subsequent `num_nalus_in_du_minus1[1] + 1` NAL units in the same access unit, 2 for the subsequent `num_nalus_in_du_minus1[2] + 1` NAL units in the same access unit, etc.
 - Otherwise, `duCpbRemovalDelayInc` is set equal to the value of `du_common_cpb_removal_delay_increment_minus1 + 1` in the picture timing SEI message, selected as specified in clause C.1, associated with access unit n .
- The nominal removal time of decoding unit m from the CPB is specified as follows, where $\text{AuNominalRemovalTime}[n]$ is the nominal removal time of access unit n :
 - If decoding unit m is the last decoding unit in access unit n , the nominal removal time of decoding unit m $\text{DuNominalRemovalTime}[m]$ is set equal to $\text{AuNominalRemovalTime}[n]$.
 - Otherwise (decoding unit m is not the last decoding unit in access unit n), the nominal removal time of decoding unit m $\text{DuNominalRemovalTime}[m]$ is derived as follows:

$$\begin{aligned} &\text{if}(\text{sub_pic_cpb_params_in_pic_timing_sei_flag}) \\ &\quad \text{DuNominalRemovalTime}[m] = \text{DuNominalRemovalTime}[m+1] - \\ &\quad \quad \text{ClockSubTick} * \text{duCpbRemovalDelayInc} \\ &\text{else} \\ &\quad \text{DuNominalRemovalTime}[m] = \text{AuNominalRemovalTime}[n] - \\ &\quad \quad \text{ClockSubTick} * \text{duCpbRemovalDelayInc} \end{aligned} \quad (\text{C-12})$$

If `SubPicHrdFlag` is equal to 0, the removal time of access unit n from the CPB is specified as follows, where $\text{AuFinalArrivalTime}[n]$ and $\text{AuNominalRemovalTime}[n]$ are the final CPB arrival time and nominal CPB removal time, respectively, of access unit n :

$$\begin{aligned} &\text{if}(\text{!low_delay_hrd_flag}[\text{HighestTid}] \parallel \text{AuNominalRemovalTime}[n] \geq \text{AuFinalArrivalTime}[n]) \\ &\quad \text{AuCpbRemovalTime}[n] = \text{AuNominalRemovalTime}[n] \\ &\text{else} \\ &\quad \text{AuCpbRemovalTime}[n] = \text{AuNominalRemovalTime}[n] + \text{ClockTick} * \\ &\quad \quad \text{Ceil}((\text{AuFinalArrivalTime}[n] - \text{AuNominalRemovalTime}[n]) \div \text{ClockTick}) \end{aligned} \quad (\text{C-13})$$

NOTE 1 – When `low_delay_hrd_flag[HighestTid]` is equal to 1 and `AuNominalRemovalTime[n]` is less than `AuFinalArrivalTime[n]`, the size of access unit `n` is so large that it prevents removal at the nominal removal time.

Otherwise (`SubPicHrdFlag` is equal to 1), the removal time of decoding unit `m` from the CPB is specified as follows:

```

if( !low_delay_hrd_flag[ HighestTid ] || DuNominalRemovalTime[ m ] >= DuFinalArrivalTime[ m ]
)
    DuCpbRemovalTime[ m ] = DuNominalRemovalTime[ m ]
else
    DuCpbRemovalTime[ m ] = DuFinalArrivalTime[ m ]

```

(C-14)

NOTE 2 – When `low_delay_hrd_flag[HighestTid]` is equal to 1 and `DuNominalRemovalTime[m]` is less than `DuFinalArrivalTime[m]`, the size of decoding unit `m` is so large that it prevents removal at the nominal removal time.

If `SubPicHrdFlag` is equal to 0, at the CPB removal time of access unit `n`, the access unit is instantaneously decoded.

Otherwise (`SubPicHrdFlag` is equal to 1), at the CPB removal time of decoding unit `m`, the decoding unit is instantaneously decoded, and when decoding unit `m` is the last decoding unit of access unit `n`, the following applies:

- Picture `n` is considered as decoded.
- The final CPB arrival time of access unit `n`, i.e., `AuFinalArrivalTime[n]`, is set equal to the final CPB arrival time of the last decoding unit in access unit `n`, i.e., `DuFinalArrivalTime[m]`.
- The nominal CPB removal time of access unit `n`, i.e., `AuNominalRemovalTime[n]`, is set equal to the nominal CPB removal time of the last decoding unit in access unit `n`, i.e., `DuNominalRemovalTime[m]`.
- The CPB removal time of access unit `n`, i.e., `AuCpbRemovalTime[m]`, is set equal to the CPB removal time of the last decoding unit in access unit `n`, i.e., `DuCpbRemovalTime[m]`.

C.3 Operation of the decoded picture buffer

C.3.1 General

The specifications in this clause apply independently to each set of decoded picture buffer (DPB) parameters selected as specified in clause C.1.

The decoded picture buffer contains picture storage buffers. Each of the picture storage buffers may contain a decoded picture that is marked as "used for reference" or is held for future output. The processes specified in clauses C.3.2, C.3.3, C.3.4 and C.3.5 are sequentially applied as specified below.

C.3.2 Removal of pictures from the DPB before decoding of the current picture

The removal of pictures from the DPB before decoding of the current picture (but after parsing the slice header of the first slice of the current picture) happens instantaneously at the CPB removal time of the first decoding unit of access unit `n` (containing the current picture) and proceeds as follows:

- The decoding process for RPS as specified in clause 8.3.2 is invoked.
- When the current picture is an IRAP picture with `NoRasOutputFlag` equal to 1 that is not picture 0, the following ordered steps are applied:
 1. The variable `NoOutputOfPriorPicsFlag` is derived for the decoder under test as follows:
 - If the current picture is a CRA picture, `NoOutputOfPriorPicsFlag` is set equal to 1 (regardless of the value of `no_output_of_prior_pics_flag`).
 - Otherwise, if the value of `pic_width_in_luma_samples`, `pic_height_in_luma_samples`, `chroma_format_idc`, `separate_colour_plane_flag`, `bit_depth_luma_minus8`, `bit_depth_chroma_minus8` or `sps_max_dec_pic_buffering_minus1[HighestTid]` derived from the active SPS is different from the value of `pic_width_in_luma_samples`, `pic_height_in_luma_samples`, `chroma_format_idc`, `separate_colour_plane_flag`, `bit_depth_luma_minus8`, `bit_depth_chroma_minus8` or `sps_max_dec_pic_buffering_minus1[HighestTid]`, respectively, derived from the SPS active for the preceding picture, `NoOutputOfPriorPicsFlag` may (but should not) be set to 1 by the decoder under test, regardless of the value of `no_output_of_prior_pics_flag`.

NOTE – Although setting `NoOutputOfPriorPicsFlag` equal to `no_output_of_prior_pics_flag` is preferred under these conditions, the decoder under test is allowed to set `NoOutputOfPriorPicsFlag` to 1 in this case.
 - Otherwise, `NoOutputOfPriorPicsFlag` is set equal to `no_output_of_prior_pics_flag`.

2. The value of NoOutputOfPriorPicsFlag derived for the decoder under test is applied for the HRD, such that when the value of NoOutputOfPriorPicsFlag is equal to 1, all picture storage buffers in the DPB are emptied without output of the pictures they contain, and the DPB fullness is set equal to 0.
- When both of the following conditions are true for any pictures k in the DPB, all such pictures k in the DPB are removed from the DPB:
 - picture k is marked as "unused for reference".
 - picture k has PicOutputFlag equal to 0 or its DPB output time is less than or equal to the CPB removal time of the first decoding unit (denoted as decoding unit m) of the current picture n; i.e., DpbOutputTime[k] is less than or equal to DuCpbRemovalTime[m].
 - For each picture that is removed from the DPB, the DPB fullness is decremented by one.

C.3.3 Picture output

The processes specified in this clause happen instantaneously at the CPB removal time of access unit n, AuCpbRemovalTime[n].

When picture n has PicOutputFlag equal to 1, its DPB output time DpbOutputTime[n] is derived as follows, where the variable firstPicInBufferingPeriodFlag is equal to 1 if access unit n is the first access unit of a buffering period and 0 otherwise:

```

if( !SubPicHrdFlag ) {
    DpbOutputTime[ n ] = AuCpbRemovalTime[ n ] + ClockTick * picDpbOutputDelay      (C-15)
    if( firstPicInBufferingPeriodFlag )
        DpbOutputTime[ n ] -= ClockTick * DpbDelayOffset
    } else
        DpbOutputTime[ n ] = AuCpbRemovalTime[ n ] + ClockSubTick * picSptDpbOutputDuDelay

```

where picDpbOutputDelay is the value of pic_dpb_output_delay in the picture timing SEI message associated with access unit n, and picSptDpbOutputDuDelay is the value of pic_spt_dpb_output_du_delay, when present, in the decoding unit information SEI messages associated with access unit n, or the value of pic_dpb_output_du_delay in the picture timing SEI message associated with access unit n when there is no decoding unit information SEI message associated with access unit n or no decoding unit information SEI message associated with access unit n has pic_spt_dpb_output_du_delay present.

NOTE – When the syntax element pic_spt_dpb_output_du_delay is not present in any decoding unit information SEI message associated with access unit n, the value is inferred to be equal to pic_dpb_output_du_delay in the picture timing SEI message associated with access unit n.

The output of the current picture is specified as follows:

- If PicOutputFlag is equal to 1 and DpbOutputTime[n] is equal to AuCpbRemovalTime[n], the current picture is output.
- Otherwise, if PicOutputFlag is equal to 0, the current picture is not output, but will be stored in the DPB as specified in clause C.3.4.
- Otherwise (PicOutputFlag is equal to 1 and DpbOutputTime[n] is greater than AuCpbRemovalTime[n]), the current picture is output later and will be stored in the DPB (as specified in clause C.3.4) and is output at time DpbOutputTime[n] unless indicated not to be output by NoOutputOfPriorPicsFlag equal to 1.

When output, the picture is cropped, using the conformance cropping window specified in the active SPS for the picture.

When picture n is a picture that is output and is not the last picture of the bitstream that is output, the value of the variable DpbOutputInterval[n] is derived as follows:

$$\text{DpbOutputInterval}[n] = \text{DpbOutputTime}[\text{nextPicInOutputOrder}] - \text{DpbOutputTime}[n] \quad (\text{C-16})$$

where nextPicInOutputOrder is the picture that follows picture n in output order and has PicOutputFlag equal to 1.

C.3.4 Current decoded picture marking and storage

The current decoded picture after the invocation of the in-loop filter process as specified in clause 8.7 is stored in the DPB in an empty picture storage buffer, the DPB fullness is incremented by one. When TwoVersionsOfCurrDecPicFlag is equal to 0 and pps_curr_pic_ref_enabled_flag is equal to 1, this picture is marked as "used for long-term reference". After all the slices of the current picture have been decoded, this picture is marked as "used for short-term reference".

When TwoVersionsOfCurrDecPicFlag is equal to 1, the current decoded picture before the invocation of the in-loop filter process as specified in clause 8.7 is stored in the DPB in an empty picture storage buffer, the DPB fullness is incremented by one, and this picture is marked as "used for long-term reference".

NOTE – Unless more memory than required by the level limit is available for storage of decoded pictures, decoders should start storing decoded parts of the current picture into the DPB when the first slice segment is decoded and continue storing more decoded samples as the decoding process proceeds.

C.3.5 Removal of pictures from the DPB after decoding of the current picture

When TwoVersionsOfCurrDecPicFlag is equal to 1, immediately after decoding of the current picture, at the CPB removal time of the last decoding unit of access unit n (containing the current picture), the current decoded picture before the invocation of the in-loop filter process as specified in clause 8.7 is removed from the DPB, and the DPB fullness is decremented by one.

C.4 Bitstream conformance

A bitstream of coded data conforming to this Specification shall fulfil all requirements specified in this clause.

The bitstream shall be constructed according to the syntax, semantics and constraints specified in this Specification outside of this annex.

The first coded picture in a bitstream shall be an IRAP picture, i.e., an IDR picture, a CRA picture or a BLA picture.

The bitstream is tested by the HRD for conformance as specified in clause C.1.

For each current picture, let the variables maxPicOrderCnt and minPicOrderCnt be set equal to the maximum and the minimum, respectively, of the PicOrderCntVal values of the following pictures:

- The current picture.
- The previous picture in decoding order that has TemporalId equal to 0 and that is not a RASL, RADL, or SLNR picture.
- The short-term reference pictures in the RPS of the current picture.
- All pictures n that have PicOutputFlag equal to 1, AuCpbRemovalTime[n] less than AuCpbRemovalTime[currPic] and DpbOutputTime[n] greater than or equal to AuCpbRemovalTime[currPic], where currPic is the current picture.

All of the following conditions shall be fulfilled for each of the bitstream conformance tests:

1. For each access unit n, with n greater than 0, associated with a buffering period SEI message, let the variable deltaTime90k[n] be specified as follows:

$$\text{deltaTime90k}[n] = 90\,000 * (\text{AuNominalRemovalTime}[n] - \text{AuFinalArrivalTime}[n-1])$$

(C-17)

The value of InitCpbRemovalDelay[SchedSelIdx] is constrained as follows:

- If cbr_flag[SchedSelIdx] is equal to 0, the following condition shall be true:

$$\text{InitCpbRemovalDelay}[\text{SchedSelIdx}] \leq \text{Ceil}(\text{deltaTime90k}[n])$$

(C-18)

- Otherwise (cbr_flag[SchedSelIdx] is equal to 1), the following condition shall be true:

$$\text{Floor}(\text{deltaTime90k}[n]) \leq \text{InitCpbRemovalDelay}[\text{SchedSelIdx}] \leq \text{Ceil}(\text{deltaTime90k}[n])$$

(C-19)

NOTE 1 – The exact number of bits in the CPB at the removal time of each picture may depend on which buffering period SEI message is selected to initialize the HRD. Encoders must take this into account to ensure that all specified constraints must be obeyed regardless of which buffering period SEI message is selected to initialize the HRD, as the HRD may be initialized at any one of the buffering period SEI messages.

2. A CPB overflow is specified as the condition in which the total number of bits in the CPB is greater than the CPB size. The CPB shall never overflow.
3. When low_delay_hrd_flag[HighestTid] is equal to 0, the CPB shall never underflow. A CPB underflow is specified as follows:

- If SubHrdFlag is equal to 0, a CPB underflow is specified as the condition in which the nominal CPB removal time of access unit n $AuNominalRemovalTime[n]$ is less than the final CPB arrival time of access unit n $AuFinalArrivalTime[n]$ for at least one value of n .
 - Otherwise (SubPicHrdFlag is equal to 1), a CPB underflow is specified as the condition in which the nominal CPB removal time of decoding unit m $DuNominalRemovalTime[m]$ is less than the final CPB arrival time of decoding unit m $DuFinalArrivalTime[m]$ for at least one value of m .
4. When SubPicHrdFlag is equal to 1, $low_delay_hrd_flag[HighestTid]$ is equal to 1 and the nominal removal time of a decoding unit m of access unit n is less than the final CPB arrival time of decoding unit m (i.e., $DuNominalRemovalTime[m] < DuFinalArrivalTime[m]$), the nominal removal time of access unit n shall be less than the final CPB arrival time of access unit n (i.e., $AuNominalRemovalTime[n] < AuFinalArrivalTime[n]$).
 5. The nominal removal times of pictures from the CPB (starting from the second picture in decoding order) shall satisfy the constraints on $AuNominalRemovalTime[n]$ and $AuCpbRemovalTime[n]$ expressed in clauses A.4.1 through A.4.2.
 6. For each current picture, after invocation of the process for removal of pictures from the DPB as specified in clause C.3.2, the number of decoded pictures in the DPB, including all pictures n that are marked as "used for reference", or that have $PicOutputFlag$ equal to 1 and $AuCpbRemovalTime[n]$ less than $AuCpbRemovalTime[currPic]$, where $currPic$ is the current picture, shall be less than or equal to $sps_max_dec_pic_buffering_minus1[HighestTid]$.
 7. All reference pictures shall be present in the DPB when needed for prediction. Each picture that has $PicOutputFlag$ equal to 1 shall be present in the DPB at its DPB output time unless it is removed from the DPB before its output time by one of the processes specified in clause C.3.
 8. For each current picture that is not an IRAP picture with $NoRaslOutputFlag$ equal to 1, the value of $maxPicOrderCnt - minPicOrderCnt$ shall be less than $MaxPicOrderCntLsb / 2$.
 9. The value of $DpbOutputInterval[n]$ as given by Equation C-16, which is the difference between the output time of a picture and that of the first picture following it in output order and having $PicOutputFlag$ equal to 1, shall satisfy the constraint expressed in clause A.4.1 for the profile, tier and level specified in the bitstream using the decoding process specified in clauses 2 through 10.
 10. For each current picture, when $sub_pic_cpb_params_in_pic_timing_sei_flag$ is equal to 1, let $tmpCpbRemovalDelaySum$ be derived as follows:

$$\begin{aligned}
 &tmpCpbRemovalDelaySum = 0 \\
 &\text{for}(i = 0; i < num_decoding_units_minus1; i++) \\
 &\quad tmpCpbRemovalDelaySum += du_cpb_removal_delay_increment_minus1[i] + 1
 \end{aligned} \tag{C-20}$$

The value of $ClockSubTick * tmpCpbRemovalDelaySum$ shall be equal to the difference between the nominal CPB removal time of the current access unit and the nominal CPB removal time of the first decoding unit in the current access unit in decoding order.

11. For any two pictures m and n in the same CVS, when $DpbOutputTime[m]$ is greater than $DpbOutputTime[n]$, the $PicOrderCntVal$ of picture m shall be greater than the $PicOrderCntVal$ of picture n .

NOTE 2 – All pictures of an earlier CVS in decoding order that are output are output before any pictures of a later CVS in decoding order. Within any particular CVS, the pictures that are output are output in increasing $PicOrderCntVal$ order.

C.5 Decoder conformance

C.5.1 General

A decoder conforming to this Specification shall fulfil all requirements specified in this clause.

A decoder claiming conformance to a specific profile, tier and level shall be able to successfully decode all bitstreams that conform to the bitstream conformance requirements specified in clause C.4, in the manner specified in Annex A, provided that all VPSs, SPSs and PPSs referred to in the VCL NAL units and appropriate buffering period, picture timing and decoding unit information SEI messages are conveyed to the decoder, in a timely manner, either in the bitstream (by non-VCL NAL units), or by external means not specified in this Specification.

When a bitstream contains syntax elements that have values that are specified as reserved and it is specified that decoders shall ignore values of the syntax elements or NAL units containing the syntax elements having the reserved values, and

the bitstream is otherwise conforming to this Specification, a conforming decoder shall decode the bitstream in the same manner as it would decode a conforming bitstream and shall ignore the syntax elements or the NAL units containing the syntax elements having the reserved values as specified.

There are two types of conformance that can be claimed by a decoder: output timing conformance and output order conformance.

To check conformance of a decoder, test bitstreams conforming to the claimed profile, tier and level, as specified in clause C.4 are delivered by a hypothetical stream scheduler (HSS) both to the HRD and to the decoder under test (DUT). All cropped decoded pictures output by the HRD shall also be output by the DUT, each cropped decoded picture output by the DUT shall be a picture with PicOutputFlag equal to 1, and, for each such cropped decoded picture output by the DUT, the values of all samples that are output shall be equal to the values of the samples produced by the specified decoding process.

For output timing decoder conformance, the HSS operates as described above, with delivery schedules selected only from the subset of values of SchedSelIdx for which the bit rate and CPB size are restricted as specified in Annex A for the specified profile, tier and level or with "interpolated" delivery schedules as specified below for which the bit rate and CPB size are restricted as specified in Annex A. The same delivery schedule is used for both the HRD and the DUT.

When the HRD parameters and the buffering period SEI messages are present with cpb_cnt_minus1[HighestTid] greater than 0, the decoder shall be capable of decoding the bitstream as delivered from the HSS operating using an "interpolated" delivery schedule specified as having peak bit rate r , CPB size $c(r)$ and initial CPB removal delay $(f(r) \div r)$ as follows:

$$\alpha = (r - \text{BitRate}[\text{SchedSelIdx} - 1]) \div (\text{BitRate}[\text{SchedSelIdx}] - \text{BitRate}[\text{SchedSelIdx} - 1]), \quad (\text{C-21})$$

$$c(r) = \alpha * \text{CpbSize}[\text{SchedSelIdx}] + (1 - \alpha) * \text{CpbSize}[\text{SchedSelIdx} - 1], \quad (\text{C-22})$$

$$f(r) = \alpha * \text{InitCpbRemovalDelay}[\text{SchedSelIdx}] * \text{BitRate}[\text{SchedSelIdx}] + (1 - \alpha) * \text{InitCpbRemovalDelay}[\text{SchedSelIdx} - 1] * \text{BitRate}[\text{SchedSelIdx} - 1] \quad (\text{C-23})$$

for any SchedSelIdx > 0 and r such that $\text{BitRate}[\text{SchedSelIdx} - 1] \leq r \leq \text{BitRate}[\text{SchedSelIdx}]$ such that r and $c(r)$ are within the limits as specified in Annex A for the maximum bit rate and buffer size for the specified profile, tier and level.

NOTE 1 – InitCpbRemovalDelay[SchedSelIdx] can be different from one buffering period to another and have to be re-calculated.

For output timing decoder conformance, an HRD as described above is used and the timing (relative to the delivery time of the first bit) of picture output is the same for both the HRD and the DUT up to a fixed delay.

For output order decoder conformance, the following applies:

- The HSS delivers the bitstream BitstreamToDecode to the DUT "by demand" from the DUT, meaning that the HSS delivers bits (in decoding order) only when the DUT requires more bits to proceed with its processing.
NOTE 2 – This means that for this test, the coded picture buffer of the DUT could be as small as the size of the largest decoding unit.
- A modified HRD as described below is used, and the HSS delivers the bitstream to the HRD by one of the schedules specified in the bitstream BitstreamToDecode such that the bit rate and CPB size are restricted as specified in Annex A. The order of pictures output shall be the same for both the HRD and the DUT.
- The HRD CPB size is given by CpbSize[SchedSelIdx] as specified in clause E.3.3, where SchedSelIdx and the HRD parameters are selected as specified in clause C.1. The DPB size is given by $\text{sps_max_dec_pic_buffering_minus1}[\text{HighestTid}] + 1$. Removal time from the CPB for the HRD is the final bit arrival time and decoding is immediate. The operation of the DPB of this HRD is as described in clauses C.5.2 through C.5.2.3.

C.5.2 Operation of the output order DPB

C.5.2.1 General

The decoded picture buffer contains picture storage buffers. Each of the picture storage buffers contains a decoded picture that is marked as "used for reference" or is held for future output. The process for output and removal of pictures from the DPB before decoding of the current picture as specified in clause C.5.2.2 is invoked, the invocation of the process for current decoded picture marking and storage as specified in clause C.3.4, further followed by the invocation of the process for removal of pictures from the DPB after decoding of the current picture as specified in clause C.3.5, and finally followed by the invocation of the process for additional bumping as specified in clause C.5.2.3. The "bumping" process is specified in clause C.5.2.4 and is invoked as specified in clauses C.5.2.2 and C.5.2.3.

C.5.2.2 Output and removal of pictures from the DPB

The output and removal of pictures from the DPB before the decoding of the current picture (but after parsing the slice header of the first slice of the current picture) happens instantaneously when the first decoding unit of the access unit containing the current picture is removed from the CPB and proceeds as follows:

- The decoding process for RPS as specified in clause 8.3.2 is invoked.
- If the current picture is an IRAP picture with NoRaslOutputFlag equal to 1 that is not picture 0, the following ordered steps are applied:
 1. The variable NoOutputOfPriorPicsFlag is derived for the decoder under test as follows:
 - If the current picture is a CRA picture, NoOutputOfPriorPicsFlag is set equal to 1 (regardless of the value of no_output_of_prior_pics_flag).
 - Otherwise, if the value of pic_width_in_luma_samples, pic_height_in_luma_samples, chroma_format_idc, separate_colour_plane_flag, bit_depth_luma_minus8, bit_depth_chroma_minus8 or sps_max_dec_pic_buffering_minus1[HighestTid] derived from the active SPS is different from the value of pic_width_in_luma_samples, pic_height_in_luma_samples, chroma_format_idc, separate_colour_plane_flag, bit_depth_luma_minus8, bit_depth_chroma_minus8 or sps_max_dec_pic_buffering_minus1[HighestTid], respectively, derived from the SPS active for the preceding picture, NoOutputOfPriorPicsFlag may (but should not) be set to 1 by the decoder under test, regardless of the value of no_output_of_prior_pics_flag.

NOTE – Although setting NoOutputOfPriorPicsFlag equal to no_output_of_prior_pics_flag is preferred under these conditions, the decoder under test is allowed to set NoOutputOfPriorPicsFlag to 1 in this case.
 - Otherwise, NoOutputOfPriorPicsFlag is set equal to no_output_of_prior_pics_flag.
 2. The value of NoOutputOfPriorPicsFlag derived for the decoder under test is applied for the HRD as follows:
 - If NoOutputOfPriorPicsFlag is equal to 1, all picture storage buffers in the DPB are emptied without output of the pictures they contain and the DPB fullness is set equal to 0.
 - Otherwise (NoOutputOfPriorPicsFlag is equal to 0), all picture storage buffers containing a picture that is marked as "not needed for output" and "unused for reference" are emptied (without output) and all non-empty picture storage buffers in the DPB are emptied by repeatedly invoking the "bumping" process specified in clause C.5.2.4 and the DPB fullness is set equal to 0.
- Otherwise (the current picture is not an IRAP picture with NoRaslOutputFlag equal to 1), all picture storage buffers containing a picture which are marked as "not needed for output" and "unused for reference" are emptied (without output). For each picture storage buffer that is emptied, the DPB fullness is decremented by one. When one or more of the following conditions are true, the "bumping" process specified in clause C.5.2.4 is invoked repeatedly while further decrementing the DPB fullness by one for each additional picture storage buffer that is emptied, until none of the following conditions are true:
 - The number of pictures in the DPB that are marked as "needed for output" is greater than sps_max_num_reorder_pics[HighestTid].
 - sps_max_latency_increase_plus1[HighestTid] is not equal to 0 and there is at least one picture in the DPB that is marked as "needed for output" for which the associated variable PicLatencyCount is greater than or equal to SpsMaxLatencyPictures[HighestTid].
 - The number of pictures in the DPB is greater than or equal to sps_max_dec_pic_buffering_minus1[HighestTid] + 1 – TwoVersionsOfCurrDecPicFlag.

C.5.2.3 Additional bumping

The processes specified in this clause happen instantaneously when the last decoding unit of access unit n containing the current picture is removed from the CPB.

When the current picture has PicOutputFlag equal to 1, for each picture in the DPB that is marked as "needed for output" and follows the current picture in output order, the associated variable PicLatencyCount is set equal to PicLatencyCount + 1.

The following applies:

- If the current decoded picture has PicOutputFlag equal to 1, it is marked as "needed for output" and its associated variable PicLatencyCount is set equal to 0.
- Otherwise (the current decoded picture has PicOutputFlag equal to 0), it is marked as "not needed for output".

When one or more of the following conditions are true, the "bumping" process specified in clause C.5.2.4 is invoked repeatedly until none of the following conditions are true:

- The number of pictures in the DPB that are marked as "needed for output" is greater than `sps_max_num_reorder_pics[HighestTid]`.
- `sps_max_latency_increase_plus1[HighestTid]` is not equal to 0 and there is at least one picture in the DPB that is marked as "needed for output" for which the associated variable `PicLatencyCount` that is greater than or equal to `SpsMaxLatencyPictures[HighestTid]`.

C.5.2.4 "Bumping" process

The "bumping" process consists of the following ordered steps:

1. The picture that is first for output is selected as the one having the smallest value of `PicOrderCntVal` of all pictures in the DPB marked as "needed for output".
2. The picture is cropped, using the conformance cropping window specified in the active SPS for the picture, the cropped picture is output, and the picture is marked as "not needed for output".
3. When the picture storage buffer that included the picture that was cropped and output contains a picture marked as "unused for reference", the picture storage buffer is emptied.

NOTE – For any two pictures `picA` and `picB` that belong to the same CVS and are output by the "bumping process", when `picA` is output earlier than `picB`, the value of `PicOrderCntVal` of `picA` is less than the value of `PicOrderCntVal` of `picB`.

Annex D

Supplemental enhancement information

(This annex forms an integral part of this Recommendation | International Standard.)

D.1 General

This annex specifies syntax and semantics for SEI message payloads.

SEI messages assist in processes related to decoding, display or other purposes. However, SEI messages are not required for constructing the luma or chroma samples by the decoding process. Conforming decoders are not required to process this information for output order conformance to this Specification (see Annex C and clause F.13 for the specification of conformance). Some SEI message information is required to check bitstream conformance and for output timing decoder conformance.

In clause C.5.2 and in clause F.13 including its subclauses, specification for presence of SEI messages are also satisfied when those messages (or some subset of them) are conveyed to decoders (or to the HRD) by other means not specified in this Specification. When present in the bitstream, SEI messages shall obey the syntax and semantics specified in clause 7.3.5 and this annex. When the content of an SEI message is conveyed for the application by some means other than presence within the bitstream, the representation of the content of the SEI message is not required to use the same syntax specified in this annex. For the purpose of counting bits, only the appropriate bits that are actually present in the bitstream are counted.

D.2 SEI payload syntax

D.2.1 General SEI message syntax

sei_payload(payloadType, payloadSize) {	Descriptor
if(nal_unit_type == PREFIX_SEI_NUT)	
if(payloadType == 0)	
buffering_period(payloadSize)	
else if(payloadType == 1)	
pic_timing(payloadSize)	
else if(payloadType == 2)	
pan_scan_rect(payloadSize)	
else if(payloadType == 3)	
filler_payload(payloadSize)	
else if(payloadType == 4)	
user_data_registered_itu_t35(payloadSize)	
else if(payloadType == 5)	
user_data_unregistered(payloadSize)	
else if(payloadType == 6)	
recovery_point(payloadSize)	
else if(payloadType == 9)	
scene_info(payloadSize)	
else if(payloadType == 15)	
picture_snapshot(payloadSize)	
else if(payloadType == 16)	
progressive_refinement_segment_start(payloadSize)	
else if(payloadType == 17)	
progressive_refinement_segment_end(payloadSize)	
else if(payloadType == 19)	
film_grain_characteristics(payloadSize)	
else if(payloadType == 22)	
post_filter_hint(payloadSize)	
else if(payloadType == 23)	
tone_mapping_info(payloadSize)	
else if(payloadType == 45)	

frame_packing_arrangement(payloadSize)	
else if(payloadType == 47)	
display_orientation(payloadSize)	
else if(payloadType == 56)	
green_metadata(payloadsize) /* specified in ISO/IEC 23001-11 */	
else if(payloadType == 128)	
structure_of_pictures_info(payloadSize)	
else if(payloadType == 129)	
active_parameter_sets(payloadSize)	
else if(payloadType == 130)	
decoding_unit_info(payloadSize)	
else if(payloadType == 131)	
temporal_sub_layer_zero_idx(payloadSize)	
else if(payloadType == 133)	
scalable_nesting(payloadSize)	
else if(payloadType == 134)	
region_refresh_info(payloadSize)	
else if(payloadType == 135)	
no_display(payloadSize)	
else if(payloadType == 136)	
time_code(payloadSize)	
else if(payloadType == 137)	
mastering_display_colour_volume(payloadSize)	
else if(payloadType == 138)	
segmented_rect_frame_packing_arrangement(payloadSize)	
else if(payloadType == 139)	
temporal_motion_constrained_tile_sets(payloadSize)	
else if(payloadType == 140)	
chroma_resampling_filter_hint(payloadSize)	
else if(payloadType == 141)	
knee_function_info(payloadSize)	
else if(payloadType == 142)	
colour_remapping_info(payloadSize)	
else if(payloadType == 143)	
deinterlaced_field_identification(payloadSize)	
else if(payloadType == 144)	
content_light_level_info(payloadSize)	
else if(payloadType == 145)	
dependent_rap_indication(payloadSize)	
else if(payloadType == 146)	
coded_region_completion(payloadSize)	
else if(payloadType == 147)	
alternative_transfer_characteristics(payloadSize)	
else if(payloadType == 148)	
ambient_viewing_environment(payloadSize)	
else if(payloadType == 149)	
content_colour_volume(payloadSize)	
else if(payloadType == 150)	
equirectangular_projection(payloadSize)	
else if(payloadType == 151)	
cubemap_projection(payloadSize)	
else if(payloadType == 152)	
fisheye_video_info(payloadSize)	

else if(payloadType == 154)	
sphere_rotation(payloadSize)	
else if(payloadType == 155)	
regionwise_packing(payloadSize)	
else if(payloadType == 156)	
omni_viewport(payloadSize)	
else if(payloadType == 157)	
regional_nesting(payloadSize)	
else if(payloadType == 158)	
mcts_extraction_info_sets(payloadSize)	
else if(payloadType == 159)	
mcts_extraction_info_nesting(payloadSize)	
else if(payloadType == 160)	
layers_not_present(payloadSize) /* specified in Annex F */	
else if(payloadType == 161)	
inter_layer_constrained_tile_sets(payloadSize) /* specified in Annex F */	
else if(payloadType == 162)	
bsp_nesting(payloadSize) /* specified in Annex F */	
else if(payloadType == 163)	
bsp_initial_arrival_time(payloadSize) /* specified in Annex F */	
else if(payloadType == 164)	
sub_bitstream_property(payloadSize) /* specified in Annex F */	
else if(payloadType == 165)	
alpha_channel_info(payloadSize) /* specified in Annex F */	
else if(payloadType == 166)	
overlay_info(payloadSize) /* specified in Annex F */	
else if(payloadType == 167)	
temporal_mv_prediction_constraints(payloadSize) /* specified in Annex F */	
else if(payloadType == 168)	
frame_field_info(payloadSize) /* specified in Annex F */	
else if(payloadType == 176)	
three_dimensional_reference_displays_info(payloadSize) /* specified in Annex G */	
else if(payloadType == 177)	
depth_representation_info(payloadSize) /* specified in Annex G */	
else if(payloadType == 178)	
multiview_scene_info(payloadSize) /* specified in Annex G */	
else if(payloadType == 179)	
multiview_acquisition_info(payloadSize) /* specified in Annex G */	
else if(payloadType == 180)	
multiview_view_position(payloadSize) /* specified in Annex G */	
else if(payloadType == 181)	
alternative_depth_info(payloadSize) /* specified in Annex I */	
else if(payloadType == 200)	
sei_manifest(payloadSize)	
else if(payloadType == 201)	
sei_prefix_indication(payloadSize)	
else if(payloadType == 202)	
annotated_regions(payloadSize)	
else if(payloadType == 205)	
shutter_interval_info(payloadSize)	
else	
reserved_sei_message(payloadSize)	

else /* nal_unit_type == SUFFIX_SEI_NUT */	
if(payloadType == 3)	
filler_payload(payloadSize)	
else if(payloadType == 4)	
user_data_registered_itu_t35(payloadSize)	
else if(payloadType == 5)	
user_data_unregistered(payloadSize)	
else if(payloadType == 17)	
progressive_refinement_segment_end(payloadSize)	
else if(payloadType == 22)	
post_filter_hint(payloadSize)	
else if(payloadType == 132)	
decoded_picture_hash(payloadSize)	
else if(payloadType == 146)	
coded_region_completion(payloadSize)	
else	
reserved_sei_message(payloadSize)	
if(more_data_in_payload()) {	
if(payload_extension_present())	
reserved_payload_extension_data	u(v)
payload_bit_equal_to_one /* equal to 1 */	f(1)
while(!byte_aligned())	
payload_bit_equal_to_zero /* equal to 0 */	f(1)
}	
}	

D.2.2 Buffering period SEI message syntax

buffering_period(payloadSize) {	Descriptor
bp_seq_parameter_set_id	ue(v)
if(!sub_pic_hrd_params_present_flag)	
irap_cpb_params_present_flag	u(1)
if(irap_cpb_params_present_flag) {	
cpb_delay_offset	u(v)
dpb_delay_offset	u(v)
}	
concatenation_flag	u(1)
au_cpb_removal_delay_delta_minus1	u(v)
if(NalHrdBpPresentFlag) {	
for(i = 0; i < CpbCnt; i++) {	
nal_initial_cpb_removal_delay[i]	u(v)
nal_initial_cpb_removal_offset[i]	u(v)
if(sub_pic_hrd_params_present_flag irap_cpb_params_present_flag) {	
nal_initial_alt_cpb_removal_delay[i]	u(v)
nal_initial_alt_cpb_removal_offset[i]	u(v)
}	
}	
}	
if(VclHrdBpPresentFlag) {	
for(i = 0; i < CpbCnt; i++) {	
vcl_initial_cpb_removal_delay[i]	u(v)
vcl_initial_cpb_removal_offset[i]	u(v)
if(sub_pic_hrd_params_present_flag irap_cpb_params_present_flag) {	
vcl_initial_alt_cpb_removal_delay[i]	u(v)
vcl_initial_alt_cpb_removal_offset[i]	u(v)
}	
}	
}	
if(more_data_in_payload())	
if(payload_extension_present())	
use_alt_cpb_params_flag	u(1)
}	

D.2.3 Picture timing SEI message syntax

	Descriptor
pic_timing(payloadSize) {	
if(frame_field_info_present_flag) {	
pic_struct	u(4)
source_scan_type	u(2)
duplicate_flag	u(1)
}	
if(CpbDpbDelaysPresentFlag) {	
au_cpb_removal_delay_minus1	u(v)
pic_dpb_output_delay	u(v)
if(sub_pic_hrd_params_present_flag)	
pic_dpb_output_du_delay	u(v)
if(sub_pic_hrd_params_present_flag && sub_pic_cpb_params_in_pic_timing_sei_flag) {	
num_decoding_units_minus1	ue(v)
du_common_cpb_removal_delay_flag	u(1)
if(du_common_cpb_removal_delay_flag)	
du_common_cpb_removal_delay_increment_minus1	u(v)
for(i = 0; i <= num_decoding_units_minus1; i++) {	
num_nalus_in_du_minus1[i]	ue(v)
if(!du_common_cpb_removal_delay_flag && i < num_decoding_units_minus1)	
du_cpb_removal_delay_increment_minus1[i]	u(v)
}	
}	
}	
}	

D.2.4 Pan-scan rectangle SEI message syntax

	Descriptor
pan_scan_rect(payloadSize) {	
pan_scan_rect_id	ue(v)
pan_scan_rect_cancel_flag	u(1)
if(!pan_scan_rect_cancel_flag) {	
pan_scan_cnt_minus1	ue(v)
for(i = 0; i <= pan_scan_cnt_minus1; i++) {	
pan_scan_rect_left_offset[i]	se(v)
pan_scan_rect_right_offset[i]	se(v)
pan_scan_rect_top_offset[i]	se(v)
pan_scan_rect_bottom_offset[i]	se(v)
}	
pan_scan_rect_persistence_flag	u(1)
}	
}	

D.2.5 Filler payload SEI message syntax

	Descriptor
filler_payload(payloadSize) {	
for(k = 0; k < payloadSize; k++)	
ff_byte /* equal to 0xFF */	f(8)
}	

D.2.6 User data registered by Recommendation ITU-T T.35 SEI message syntax

	Descriptor
user_data_registered_itu_t_t35(payloadSize) {	
itu_t_t35_country_code	b(8)
if(itu_t_t35_country_code != 0xFF)	
i = 1	
else {	
itu_t_t35_country_code_extension_byte	b(8)
i = 2	
}	
do {	
itu_t_t35_payload_byte	b(8)
i++	
} while(i < payloadSize)	
}	

D.2.7 User data unregistered SEI message syntax

	Descriptor
user_data_unregistered(payloadSize) {	
uuid_iso_iec_11578	u(128)
for(i = 16; i < payloadSize; i++)	
user_data_payload_byte	b(8)
}	

D.2.8 Recovery point SEI message syntax

	Descriptor
recovery_point(payloadSize) {	
recovery_poc_cnt	se(v)
exact_match_flag	u(1)
broken_link_flag	u(1)
}	

D.2.9 Scene information SEI message syntax

scene_info(payloadSize) {	Descriptor
scene_info_present_flag	u(1)
if(scene_info_present_flag) {	
prev_scene_id_valid_flag	u(1)
scene_id	ue(v)
scene_transition_type	ue(v)
if(scene_transition_type > 3)	
second_scene_id	ue(v)
}	
}	

D.2.10 Picture snapshot SEI message syntax

picture_snapshot(payloadSize) {	Descriptor
snapshot_id	ue(v)
}	

D.2.11 Progressive refinement segment start SEI message syntax

progressive_refinement_segment_start(payloadSize) {	Descriptor
progressive_refinement_id	ue(v)
pic_order_cnt_delta	ue(v)
}	

D.2.12 Progressive refinement segment end SEI message syntax

progressive_refinement_segment_end(payloadSize) {	Descriptor
progressive_refinement_id	ue(v)
}	

D.2.13 Film grain characteristics SEI message syntax

film_grain_characteristics(payloadSize) {	Descriptor
film_grain_characteristics_cancel_flag	u(1)
if(!film_grain_characteristics_cancel_flag) {	
film_grain_model_id	u(2)
separate_colour_description_present_flag	u(1)
if(separate_colour_description_present_flag) {	
film_grain_bit_depth_luma_minus8	u(3)
film_grain_bit_depth_chroma_minus8	u(3)
film_grain_full_range_flag	u(1)
film_grain_colour_primaries	u(8)
film_grain_transfer_characteristics	u(8)
film_grain_matrix_coeffs	u(8)
}	
blending_mode_id	u(2)
log2_scale_factor	u(4)
for(c = 0; c < 3; c++)	
comp_model_present_flag[c]	u(1)
for(c = 0; c < 3; c++)	
if(comp_model_present_flag[c]) {	
num_intensity_intervals_minus1[c]	u(8)
num_model_values_minus1[c]	u(3)
for(i = 0; i <= num_intensity_intervals_minus1[c]; i++) {	
intensity_interval_lower_bound[c][i]	u(8)
intensity_interval_upper_bound[c][i]	u(8)
for(j = 0; j <= num_model_values_minus1[c]; j++)	
comp_model_value[c][i][j]	se(v)
}	
}	
film_grain_characteristics_persistence_flag	u(1)
}	
}	

D.2.14 Post-filter hint SEI message syntax

post_filter_hint(payloadSize) {	Descriptor
filter_hint_size_y	ue(v)
filter_hint_size_x	ue(v)
filter_hint_type	u(2)
for(cIdx = 0; cIdx < (chroma_format_idc == 0 ? 1 : 3); cIdx++)	
for(cy = 0; cy < filter_hint_size_y; cy++)	
for(cx = 0; cx < filter_hint_size_x; cx++)	
filter_hint_value[cIdx][cy][cx]	se(v)
}	

D.2.15 Tone mapping information SEI message syntax

tone_mapping_info(payloadSize) {	Descriptor
tone_map_id	ue(v)
tone_map_cancel_flag	u(1)
if(!tone_map_cancel_flag) {	
tone_map_persistence_flag	u(1)
coded_data_bit_depth	u(8)
target_bit_depth	u(8)
tone_map_model_id	ue(v)
if(tone_map_model_id == 0) {	
min_value	u(32)
max_value	u(32)
} else if(tone_map_model_id == 1) {	
sigmoid_midpoint	u(32)
sigmoid_width	u(32)
} else if(tone_map_model_id == 2)	
for(i = 0; i < (1 << target_bit_depth); i++)	
start_of_coded_interval[i]	u(v)
else if(tone_map_model_id == 3) {	
num_pivots	u(16)
for(i = 0; i < num_pivots; i++) {	
coded_pivot_value[i]	u(v)
target_pivot_value[i]	u(v)
}	
} else if(tone_map_model_id == 4) {	
camera_iso_speed_idc	u(8)
if(camera_iso_speed_idc == EXTENDED_ISO)	
camera_iso_speed_value	u(32)
exposure_idx_idc	u(8)
if(exposure_idx_idc == EXTENDED_ISO)	
exposure_idx_value	u(32)
exposure_compensation_value_sign_flag	u(1)
exposure_compensation_value_numerator	u(16)
exposure_compensation_value_denom_idc	u(16)
ref_screen_luminance_white	u(32)
extended_range_white_level	u(32)
nominal_black_level_code_value	u(16)
nominal_white_level_code_value	u(16)
extended_white_level_code_value	u(16)
}	
}	
}	

D.2.16 Frame packing arrangement SEI message syntax

frame_packing_arrangement(payloadSize) {	Descriptor
frame_packing_arrangement_id	ue(v)
frame_packing_arrangement_cancel_flag	u(1)
if(!frame_packing_arrangement_cancel_flag) {	
frame_packing_arrangement_type	u(7)
quincunx_sampling_flag	u(1)
content_interpretation_type	u(6)
spatial_flipping_flag	u(1)
frame0_flipped_flag	u(1)
field_views_flag	u(1)
current_frame_is_frame0_flag	u(1)
frame0_self_contained_flag	u(1)
frame1_self_contained_flag	u(1)
if(!quincunx_sampling_flag && frame_packing_arrangement_type != 5) {	
frame0_grid_position_x	u(4)
frame0_grid_position_y	u(4)
frame1_grid_position_x	u(4)
frame1_grid_position_y	u(4)
}	
frame_packing_arrangement_reserved_byte	u(8)
frame_packing_arrangement_persistence_flag	u(1)
}	
upsampled_aspect_ratio_flag	u(1)
}	

D.2.17 Display orientation SEI message syntax

display_orientation(payloadSize) {	Descriptor
display_orientation_cancel_flag	u(1)
if(!display_orientation_cancel_flag) {	
hor_flip	u(1)
ver_flip	u(1)
anticlockwise_rotation	u(16)
display_orientation_persistence_flag	u(1)
}	
}	

D.2.18 Green metadata SEI message syntax

The syntax for green metadata SEI message is specified in ISO/IEC 23001-11 (Green metadata). Green metadata facilitates reduced power consumption in decoders, encoders, displays and in media selection.

D.2.19 Structure of pictures information SEI message syntax

structure_of_pictures_info(payloadSize) {	Descriptor
sop_seq_parameter_set_id	ue(v)
num_entries_in_sop_minus1	ue(v)
for(i = 0; i <= num_entries_in_sop_minus1; i++) {	
sop_vcl_nut[i]	u(6)
sop_temporal_id[i]	u(3)
if(sop_vcl_nut[i] != IDR_W_RADL && sop_vcl_nut[i] != IDR_N_LP)	
sop_short_term_rps_idx[i]	ue(v)
if(i > 0)	
sop_poc_delta[i]	se(v)
}	
}	

D.2.20 Decoded picture hash SEI message syntax

decoded_picture_hash(payloadSize) {	Descriptor
hash_type	u(8)
for(cIdx = 0; cIdx < (chroma_format_idc == 0 ? 1 : 3); cIdx++)	
if(hash_type == 0)	
for(i = 0; i < 16; i++)	
picture_md5[cIdx][i]	b(8)
else if(hash_type == 1)	
picture_crc[cIdx]	u(16)
else if(hash_type == 2)	
picture_checksum[cIdx]	u(32)
}	

D.2.21 Active parameter sets SEI message syntax

active_parameter_sets(payloadSize) {	Descriptor
active_video_parameter_set_id	u(4)
self_contained_cvs_flag	u(1)
no_parameter_set_update_flag	u(1)
num_sps_ids_minus1	ue(v)
for(i = 0; i <= num_sps_ids_minus1; i++)	
active_seq_parameter_set_id[i]	ue(v)
for(i = vps_base_layer_internal_flag; i <= MaxLayersMinus1; i++)	
layer_sps_idx[i]	ue(v)
}	

D.2.22 Decoding unit information SEI message syntax

	Descriptor
decoding_unit_info(payloadSize) {	
decoding_unit_idx	ue(v)
if(!sub_pic_cpb_params_in_pic_timing_sei_flag)	
du_spt_cpb_removal_delay_increment	u(v)
dpb_output_du_delay_present_flag	u(1)
if(dpb_output_du_delay_present_flag)	
pic_spt_dpb_output_du_delay	u(v)
}	

D.2.23 Temporal sub-layer zero index SEI message syntax

	Descriptor
temporal_sub_layer_zero_idx(payloadSize) {	
temporal_sub_layer_zero_idx	u(8)
irap_pic_id	u(8)
}	

D.2.24 Scalable nesting SEI message syntax

	Descriptor
scalable_nesting(payloadSize) {	
bitstream_subset_flag	u(1)
nesting_op_flag	u(1)
if(nesting_op_flag) {	
default_op_flag	u(1)
nesting_num_ops_minus1	ue(v)
for(i = default_op_flag; i <= nesting_num_ops_minus1; i++) {	
nesting_max_temporal_id_plus1[i]	u(3)
nesting_op_idx[i]	ue(v)
}	
} else {	
all_layers_flag	u(1)
if(!all_layers_flag) {	
nesting_no_op_max_temporal_id_plus1	u(3)
nesting_num_layers_minus1	ue(v)
for(i = 0; i <= nesting_num_layers_minus1; i++)	
nesting_layer_id[i]	u(6)
}	
}	
while(!byte_aligned())	
nesting_zero_bit /* equal to 0 */	u(1)
do	
sei_message()	
while(more_rbsp_data())	
}	

D.2.25 Region refresh information SEI message syntax

	Descriptor
region_refresh_info(payloadSize) {	
refreshed_region_flag	u(1)
}	

D.2.26 No display SEI message syntax

	Descriptor
no_display(payloadSize) {	
}	

D.2.27 Time code SEI message syntax

	Descriptor
time_code(payloadSize) {	
num_clock_ts	u(2)
for(i = 0; i < num_clock_ts; i++) {	
clock_timestamp_flag[i]	u(1)
if(clock_timestamp_flag[i]) {	
units_field_based_flag[i]	u(1)
counting_type[i]	u(5)
full_timestamp_flag[i]	u(1)
discontinuity_flag[i]	u(1)
cnt_dropped_flag[i]	u(1)
n_frames[i]	u(9)
if(full_timestamp_flag[i]) {	
seconds_value[i] /* 0..59 */	u(6)
minutes_value[i] /* 0..59 */	u(6)
hours_value[i] /* 0..23 */	u(5)
} else {	
seconds_flag[i]	u(1)
if(seconds_flag[i]) {	
seconds_value[i] /* 0..59 */	u(6)
minutes_flag[i]	u(1)
if(minutes_flag[i]) {	
minutes_value[i] /* 0..59 */	u(6)
hours_flag[i]	u(1)
if(hours_flag[i])	
hours_value[i] /* 0..23 */	u(5)
}	
}	
}	
}	
time_offset_length[i]	u(5)
if(time_offset_length[i] > 0)	
time_offset_value[i]	i(v)
}	
}	
}	

D.2.28 Mastering display colour volume SEI message syntax

mastering_display_colour_volume(payloadSize) {	Descriptor
for(c = 0; c < 3; c++) {	
display primaries_x [c]	u(16)
display primaries_y [c]	u(16)
}	
white_point_x	u(16)
white_point_y	u(16)
max_display_mastering_luminance	u(32)
min_display_mastering_luminance	u(32)
}	

D.2.29 Segmented rectangular frame packing arrangement SEI message syntax

segmented_rect_frame_packing_arrangement(payloadSize) {	Descriptor
segmented_rect_frame_packing_arrangement_cancel_flag	u(1)
if(!segmented_rect_frame_packing_arrangement_cancel_flag) {	
segmented_rect_content_interpretation_type	u(2)
segmented_rect_frame_packing_arrangement_persistence_flag	u(1)
}	
}	

D.2.30 Temporal motion-constrained tile sets SEI message syntax

temporal_motion_constrained_tile_sets(payloadSize) {	Descriptor
mc_all_tiles_exact_sample_value_match_flag	u(1)
each_tile_one_tile_set_flag	u(1)
if(!each_tile_one_tile_set_flag) {	
limited_tile_set_display_flag	u(1)
num_sets_in_message_minus1	ue(v)
for(i = 0; i <= num_sets_in_message_minus1; i++) {	
mcts_id[i]	ue(v)
if(limited_tile_set_display_flag)	
display_tile_set_flag[i]	u(1)
num_tile_rects_in_set_minus1[i]	ue(v)
for(j = 0; j <= num_tile_rects_in_set_minus1[i]; j++) {	
top_left_tile_idx[i][j]	ue(v)
bottom_right_tile_idx[i][j]	ue(v)
}	
if(!mc_all_tiles_exact_sample_value_match_flag)	
mc_exact_sample_value_match_flag[i]	u(1)
mcts_tier_level_idc_present_flag[i]	u(1)
if(mcts_tier_level_idc_present_flag[i]) {	
mcts_tier_flag[i]	u(1)
mcts_level_idc[i]	u(8)
}	
}	
} else {	
max_mcs_tier_level_idc_present_flag	u(1)
if(mcts_max_tier_level_idc_present_flag) {	
mcts_max_tier_flag	u(1)
mcts_max_level_idc	u(8)
}	
}	
}	

D.2.31 Chroma resampling filter hint SEI message syntax

chroma_resampling_filter_hint(payloadSize) {	Descriptor
ver_chroma_filter_idc	u(8)
hor_chroma_filter_idc	u(8)
ver_filtering_field_processing_flag	u(1)
if(ver_chroma_filter_idc == 1 hor_chroma_filter_idc == 1) {	
target_format_idc	ue(v)
if(ver_chroma_filter_idc == 1) {	
num_vertical_filters	ue(v)
for(i = 0; i < num_vertical_filters; i++) {	
ver_tap_length_minus1[i]	ue(v)
for(j = 0; j <= ver_tap_length_minus1[i]; j++)	
ver_filter_coeff[i][j]	se(v)
}	
}	
if(hor_chroma_filter_idc == 1) {	
num_horizontal_filters	ue(v)
for(i = 0; i < num_horizontal_filters; i++) {	
hor_tap_length_minus1[i]	ue(v)
for(j = 0; j <= hor_tap_length_minus1[i]; j++)	
hor_filter_coeff[i][j]	se(v)
}	
}	
}	
}	

D.2.32 Knee function information SEI message syntax

knee_function_info(payloadSize) {	Descriptor
knee_function_id	ue(v)
knee_function_cancel_flag	u(1)
if(!knee_function_cancel_flag) {	
knee_function_persistence_flag	u(1)
input_d_range	u(32)
input_disp_luminance	u(32)
output_d_range	u(32)
output_disp_luminance	u(32)
num_knee_points_minus1	ue(v)
for(i = 0; i <= num_knee_points_minus1; i++) {	
input_knee_point[i]	u(10)
output_knee_point[i]	u(10)
}	
}	
}	

D.2.33 Colour remapping information SEI message syntax

colour_remapping_info(payloadSize) {	Descriptor
colour_remap_id	ue(v)
colour_remap_cancel_flag	u(1)
if(!colour_remap_cancel_flag) {	
colour_remap_persistence_flag	u(1)
colour_remap_video_signal_info_present_flag	u(1)
if(colour_remap_video_signal_info_present_flag) {	
colour_remap_full_range_flag	u(1)
colour_remap_primaries	u(8)
colour_remap_transfer_function	u(8)
colour_remap_matrix_coefficients	u(8)
}	
colour_remap_input_bit_depth	u(8)
colour_remap_output_bit_depth	u(8)
for(c = 0; c < 3; c++) {	
pre_lut_num_val_minus1[c]	u(8)
if(pre_lut_num_val_minus1[c] > 0)	
for(i = 0; i <= pre_lut_num_val_minus1[c]; i++) {	
pre_lut_coded_value[c][i]	u(v)
pre_lut_target_value[c][i]	u(v)
}	
}	
colour_remap_matrix_present_flag	u(1)
if(colour_remap_matrix_present_flag) {	
log2_matrix_denom	u(4)
for(c = 0; c < 3; c++)	
for(i = 0; i < 3; i++)	
colour_remap_coeffs[c][i]	se(v)
}	
for(c = 0; c < 3; c++) {	
post_lut_num_val_minus1[c]	u(8)
if(post_lut_num_val_minus1[c] > 0)	
for(i = 0; i <= post_lut_num_val_minus1[c]; i++) {	
post_lut_coded_value[c][i]	u(v)
post_lut_target_value[c][i]	u(v)
}	
}	
}	
}	

D.2.34 Deinterlaced field identification SEI message syntax

deinterlaced_field_indentification(payloadSize) {	Descriptor
deinterlaced_picture_source_parity_flag	u(1)
}	

D.2.35 Content light level information SEI message syntax

content_light_level_info(payloadSize) {	Descriptor
max_content_light_level	u(16)
max_pic_average_light_level	u(16)
}	

D.2.36 Dependent random access point indication SEI message syntax

dependent_rap_indication(payloadSize) {	Descriptor
}	

D.2.37 Coded region completion SEI message syntax

coded_region_completion(payloadSize) {	Descriptor
next_segment_address	ue(v)
if(next_segment_address > 0)	
independent_slice_segment_flag	u(1)
}	

D.2.38 Alternative transfer characteristics information SEI message syntax

alternative_transfer_characteristics (payloadSize) {	Descriptor
preferred_transfer_characteristics	u(8)
}	

D.2.39 Ambient viewing environment SEI message syntax

ambient_viewing_environment(payloadSize) {	Descriptor
ambient_illuminance	u(32)
ambient_light_x	u(16)
ambient_light_y	u(16)
}	

D.2.40 Content colour volume SEI message syntax

content_colour_volume(payloadSize) {	Descriptor
ccv_cancel_flag	u(1)
if(!ccv_cancel_flag) {	
ccv_persistence_flag	u(1)
ccv_primaries_present_flag	u(1)
ccv_min_luminance_value_present_flag	u(1)
ccv_max_luminance_value_present_flag	u(1)
ccv_avg_luminance_value_present_flag	u(1)
ccv_reserved_zero_2bits	u(2)
if(ccv_primaries_present_flag)	
for(c = 0; c < 3; c++) {	
ccv_primaries_x[c]	i(32)
ccv_primaries_y[c]	i(32)
}	
if(ccv_min_luminance_value_present_flag)	
ccv_min_luminance_value	u(32)
if(ccv_max_luminance_value_present_flag)	
ccv_max_luminance_value	u(32)
if(ccv_avg_luminance_value_present_flag)	
ccv_avg_luminance_value	u(32)
}	
}	

D.2.41 Syntax of omnidirectional video specific SEI messages

D.2.41.1 Equirectangular projection SEI message syntax

equirectangular_projection(payloadSize) {	Descriptor
erp_cancel_flag	u(1)
if(!erp_cancel_flag) {	
erp_persistence_flag	u(1)
erp_guard_band_flag	u(1)
erp_reserved_zero_2bits	u(2)
if(erp_guard_band_flag == 1) {	
erp_guard_band_type	u(3)
erp_left_guard_band_width	u(8)
erp_right_guard_band_width	u(8)
}	
}	
}	

D.2.41.2 Cubemap projection SEI message syntax

cubemap_projection(payloadSize) {	Descriptor
cmp_cancel_flag	u(1)
if(!cmp_cancel_flag)	
cmp_persistence_flag	u(1)
}	

D.2.41.3 Fisheye video information SEI message syntax

fisheye_video_info(payloadSize) {	Descriptor
fisheye_cancel_flag	u(1)
if(!fisheye_cancel_flag) {	
fisheye_persistence_flag	u(1)
fisheye_view_dimension_idc	u(3)
fisheye_reserved_zero_3bits	u(3)
fisheye_num_active_areas_minus1	u(8)
for(i = 0; i <= fisheye_num_active_areas_minus1; i++) {	
fisheye_circular_region_centre_x[i]	u(32)
fisheye_circular_region_centre_y[i]	u(32)
fisheye_rect_region_top[i]	u(32)
fisheye_rect_region_left[i]	u(32)
fisheye_rect_region_width[i]	u(32)
fisheye_rect_region_height[i]	u(32)
fisheye_circular_region_radius[i]	u(32)
fisheye_scene_radius[i]	u(32)
fisheye_camera_centre_azimuth[i]	i(32)
fisheye_camera_centre_elevation[i]	i(32)
fisheye_camera_centre_tilt[i]	i(32)
fisheye_camera_centre_offset_x[i]	u(32)
fisheye_camera_centre_offset_y[i]	u(32)
fisheye_camera_centre_offset_z[i]	u(32)
fisheye_field_of_view[i]	u(32)
fisheye_num_polynomial_coeffs[i]	u(16)
for(j = 0; j < fisheye_num_polynomial_coeffs[i]; j++)	
fisheye_polynomial_coeff[i][j]	i(32)
}	
}	
}	

D.2.41.4 Sphere rotation SEI message syntax

	Descriptor
sphere_rotation(payloadSize) {	
sphere_rotation_cancel_flag	u(1)
if(!sphere_rotation_cancel_flag) {	
sphere_rotation_persistence_flag	u(1)
sphere_rotation_reserved_zero_6bits	u(6)
yaw_rotation	i(32)
pitch_rotation	i(32)
roll_rotation	i(32)
}	
}	

D.2.41.5 Region-wise packing SEI message syntax

regionwise_packing(payloadSize) {	Descriptor
rwp_cancel_flag	u(1)
if(!rwp_cancel_flag) {	
rwp_persistence_flag	u(1)
constituent_picture_matching_flag	u(1)
rwp_reserved_zero_5bits	u(5)
num_packed_regions	u(8)
proj_picture_width	u(32)
proj_picture_height	u(32)
packed_picture_width	u(16)
packed_picture_height	u(16)
for(i = 0; i < num_packed_regions; i++) {	
rwp_reserved_zero_4bits[i]	u(4)
rwp_transform_type[i]	u(3)
rwp_guard_band_flag[i]	u(1)
proj_region_width[i]	u(32)
proj_region_height[i]	u(32)
proj_region_top[i]	u(32)
proj_region_left[i]	u(32)
packed_region_width[i]	u(16)
packed_region_height[i]	u(16)
packed_region_top[i]	u(16)
packed_region_left[i]	u(16)
if(rwp_guard_band_flag[i]) {	
rwp_left_guard_band_width[i]	u(8)
rwp_right_guard_band_width[i]	u(8)
rwp_top_guard_band_height[i]	u(8)
rwp_bottom_guard_band_height[i]	u(8)
rwp_guard_band_not_used_for_pred_flag[i]	u(1)
for(j = 0; j < 4; j++)	
rwp_guard_band_type[i][j]	u(3)
rwp_guard_band_reserved_zero_3bits[i]	u(3)
}	
}	
}	
}	

D.2.41.6 Omnidirectional viewport SEI message syntax

omni_viewport(payloadSize) {	Descriptor
omni_viewport_id	u(10)
omni_viewport_cancel_flag	u(1)
if(!omni_viewport_cancel_flag) {	
omni_viewport_persistence_flag	u(1)
omni_viewport_cnt_minus1	u(4)
for(i = 0; i <= omni_viewport_cnt_minus1; i++) {	
omni_viewport_azimuth_centre[i]	i(32)
omni_viewport_elevation_centre[i]	i(32)
omni_viewport_tilt_centre[i]	i(32)
omni_viewport_hor_range[i]	u(32)
omni_viewport_ver_range[i]	u(32)
}	
}	
}	

D.2.42 Regional nesting SEI message syntax

regional_nesting(payloadSize) {	Descriptor
regional_nesting_id	u(16)
regional_nesting_num_rect_regions	u(8)
for(i = 0; i < regional_nesting_num_rect_regions; i++) {	
regional_nesting_rect_region_id[i]	u(8)
regional_nesting_rect_left_offset[i]	u(16)
regional_nesting_rect_right_offset[i]	u(16)
regional_nesting_rect_top_offset[i]	u(16)
regional_nesting_rect_bottom_offset[i]	u(16)
}	
num_sei_messages_in_regional_nesting_minus1	u(8)
for(i = 0; i <= num_sei_messages_in_regional_nesting_minus1; i++) {	
num_regions_for_sei_message[i]	u(8)
for(j = 0; j < num_regions_for_sei_message[i]; j++)	
regional_nesting_sei_region_idx[i][j]	u(8)
sei_message()	
}	
}	

D.2.43 Motion-constrained tile sets extraction information sets SEI message syntax

mcts_extraction_info_sets(payloadSize) {	Descriptor
num_info_sets_minus1	ue(v)
for(i = 0; i <= num_info_sets_minus1; i++) {	
num_mcts_sets_minus1[i]	ue(v)
for(j = 0; j <= num_mcts_sets_minus1[i]; j++) {	
num_mcts_in_set_minus1[i][j]	ue(v)
for(k = 0; k <= num_mcts_in_set_minus1[i][j]; k++)	
idx_of_mcts_in_set[i][j][k]	ue(v)
}	
slice_reordering_enabled_flag[i]	u(1)
if(slice_reordering_enabled_flag[i]) {	
num_slice_segments_minus1[i]	ue(v)
for(j = 0; j <= num_slice_segments_minus1[i]; j++)	
output_slice_segment_address[i][j]	u(v)
}	
num_vps_in_info_set_minus1[i]	ue(v)
for(j = 0; j <= num_vps_in_info_set_minus1[i]; j++)	
vps_rbsp_data_length[i][j]	ue(v)
num_sps_in_info_set_minus1[i]	ue(v)
for(j = 0; j <= num_sps_in_info_set_minus1[i]; j++)	
sps_rbsp_data_length[i][j]	ue(v)
num_pps_in_info_set_minus1[i]	ue(v)
for(j = 0; j <= num_pps_in_info_set_minus1[i]; j++) {	
pps_nuh_temporal_id_plus1[i][j]	u(3)
pps_rbsp_data_length[i][j]	ue(v)
}	
while(!byte_aligned())	
mcts_alignment_bit_equal_to_zero	f(1)
for(j = 0; j <= num_vps_in_info_set_minus1[i]; j++)	
for(k = 0; k < vps_rbsp_data_length[i][j]; k++)	
vps_rbsp_data_byte[i][j][k]	u(8)
for(j = 0; j <= num_sps_in_info_set_minus1[i]; j++)	
for(k = 0; k < sps_rbsp_data_length[i][j]; k++)	
sps_rbsp_data_byte[i][j][k]	u(8)
for(j = 0; j <= num_pps_in_info_set_minus1[i]; j++)	
for(k = 0; k < pps_rbsp_data_length[i][j]; k++)	
pps_rbsp_data_byte[i][j][k]	u(8)
}	
}	

D.2.44 Motion-constrained tile sets extraction information nesting SEI message syntax

	Descriptor
mcts_extraction_info_nesting(payloadSize) {	
all_mcts_flag	u(1)
if(!all_mcts_flag) {	
num_associated_mcts_minus1	ue(v)
for(i = 0; i <= num_associated_mcts_minus1; i++)	
idx_of_associated_mcts[i]	ue(v)
}	
num_sei_messages_in_mcts_extraction_nesting_minus1	ue(v)
while(!byte_aligned())	
mcts_nesting_zero_bit /* equal to 0 */	u(1)
for(i = 0; i <= num_sei_messages_in_mcts_extraction_nesting_minus1; i++)	
sei_message()	
}	

D.2.45 SEI manifest SEI message syntax

	Descriptor
sei_manifest(payloadSize) {	
manifest_num_sei_msg_types	u(16)
for(i = 0; i < manifest_num_sei_msg_types; i++) {	
manifest_sei_payload_type[i]	u(16)
manifest_sei_description[i]	u(8)
}	
}	

D.2.46 SEI prefix indication SEI message syntax

	Descriptor
sei_prefix_indication(payloadSize) {	
prefix_sei_payload_type	u(16)
num_sei_prefix_indications_minus1	u(8)
for(i = 0; i <= num_sei_prefix_indications_minus1; i++) {	
num_bits_in_prefix_indication_minus1[i]	u(16)
for(j = 0; j <= num_bits_in_prefix_indication_minus1[i]; j++)	
sei_prefix_data_bit[i][j]	u(1)
while(!byte_aligned())	
byte_alignment_bit_equal_to_one /* equal to 1 */	f(1)
}	
}	

D.2.47 Annotated regions SEI message syntax

annotated_regions(payloadSize) {	Descriptor
ar_cancel_flag	u(1)
if(!ar_cancel_flag) {	
ar_not_optimized_for_viewing_flag	u(1)
ar_true_motion_flag	u(1)
ar_occluded_object_flag	u(1)
ar_partial_object_flag_present_flag	u(1)
ar_object_label_present_flag	u(1)
ar_object_confidence_info_present_flag	u(1)
if(ar_object_confidence_info_present_flag)	
ar_object_confidence_length_minus1	u(4)
if(ar_object_label_present_flag) {	
ar_object_label_language_present_flag	u(1)
if(ar_object_label_language_present_flag) {	
while(!byte_aligned())	
ar_bit_equal_to_zero /* equal to 0 */	f(1)
ar_object_label_language	st(v)
}	
ar_num_label_updates	ue(v)
for(i = 0; i < ar_num_label_updates; i++) {	
ar_label_idx[i]	ue(v)
ar_label_cancel_flag	u(1)
LabelAssigned[ar_label_idx[i]] = !ar_label_cancel_flag	
if(!ar_label_cancel_flag) {	
while(!byte_aligned())	
ar_bit_equal_to_zero /* equal to 0 */	f(1)
ar_label[ar_label_idx[i]]	st(v)
}	
}	
}	
ar_num_object_updates	ue(v)
for(i = 0; i < ar_num_object_updates; i++) {	
ar_object_idx[i]	ue(v)
ar_object_cancel_flag	u(1)
ObjectTracked[ar_object_idx[i]] = !ar_object_cancel_flag	
if(!ar_object_cancel_flag) {	
if(ar_object_label_present_flag) {	
ar_object_label_update_flag	u(1)
if(ar_object_label_update_flag)	
ar_object_label_idx[ar_object_idx[i]]	ue(v)
}	
ar_bounding_box_update_flag	u(1)
if(ar_bounding_box_update_flag) {	
ar_bounding_box_cancel_flag	u(1)
ObjectBoundingBoxAvail[ar_object_idx[i]] = !ar_bounding_box_cancel_flag	

if(!ar_bounding_box_cancel_flag) {	
ar_bounding_box_top [ar_object_idx[i]]	u(16)
ar_bounding_box_left [ar_object_idx[i]]	u(16)
ar_bounding_box_width [ar_object_idx[i]]	u(16)
ar_bounding_box_height [ar_object_idx[i]]	u(16)
if(ar_partial_object_flag_present_flag)	
ar_partial_object_flag [ar_object_idx[i]]	u(1)
if(ar_object_confidence_info_present_flag)	
ar_object_confidence [ar_object_idx[i]]	u(v)
}	
}	
}	
}	
}	
}	

D.2.48 Shutter interval information SEI message syntax

shutter_interval_info(payloadSize) {	Descriptor
sii_time_scale	u(32)
fixed_shutter_interval_within_clvs_flag	u(1)
if(fixed_shutter_interval_within_clvs_flag)	
sii_num_units_in_shutter_interval	u(32)
else {	
sii_max_sub_layers_minus1	u(3)
for(i = 0; i <= sii_max_sub_layers_minus1; i++)	
sub_layer_num_units_in_shutter_interval [i]	u(32)
}	
}	

D.2.49 Reserved SEI message syntax

reserved_sei_message(payloadSize) {	Descriptor
for(i = 0; i < payloadSize; i++)	
reserved_sei_message_payload_byte	b(8)
}	

D.3 SEI payload semantics

D.3.1 General SEI payload semantics

reserved_payload_extension_data shall not be present in bitstreams conforming to this version of this Specification. However, decoders conforming to this version of this Specification shall ignore the presence and value of **reserved_payload_extension_data**. When present, the length, in bits, of **reserved_payload_extension_data** is equal to $8 * \text{payloadSize} - \text{nEarlierBits} - \text{nPayloadZeroBits} - 1$, where **nEarlierBits** is the number of bits in the **sei_payload()** syntax structure that precede the **reserved_payload_extension_data** syntax element and **nPayloadZeroBits** is the number of **payload_bit_equal_to_zero** syntax elements at the end of the **sei_payload()** syntax structure.

payload_bit_equal_to_one shall be equal to 1.

payload_bit_equal_to_zero shall be equal to 0.

NOTE 1 – SEI messages with the same value of payloadType are conceptually the same SEI message regardless of whether they are contained in prefix or suffix SEI NAL units.

NOTE 2 – For SEI messages with payloadType in the range of 0 to 47, inclusive, that are specified in this Specification, the payloadType values are aligned with similar SEI messages specified in Rec. ITU-T H.264 | ISO/IEC 14496-10.

The list SingleLayerSeiList is set to consist of the payloadType values 2, 3, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 129, 131, 132, 134 to 152, inclusive, 154 to 159, inclusive, 200 to 202, inclusive, and 205.

The list VclAssociatedSeiList is set to consist of the payloadType values 2, 3, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 131, 132, 134 to 152, inclusive, 154 to 159, inclusive, 200 to 202, inclusive, and 205.

The list PicUnitRepConSeiList is set to consist of the payloadType values 0, 1, 2, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 129, 131, 132, 133, 135 to 152, inclusive, 154 to 159, inclusive, 200 to 202, inclusive, and 205.

NOTE 3 – SingleLayerSeiList consists of the payloadType values of the SEI messages specified in Annex D excluding 0 (buffering period), 1 (picture timing), 4 (user data registered by Recommendation ITU-T T.35), 5 (user data unregistered), 130 (decoding unit information) and 133 (scalable nesting). VclAssociatedSeiList consists of the payloadType values of the SEI messages that, when non-scalable-nested and contained in an SEI NAL unit, infer constraints on the NAL unit header of the SEI NAL unit on the basis of the NAL unit header of the associated VCL NAL unit. PicUnitRepConSeiList consists of the payloadType values of the SEI messages that are subject to the restriction on 8 repetitions per picture unit.

The semantics and persistence scope for each SEI message are specified in the semantics specification for each particular SEI message.

NOTE 4 – Persistence information for SEI messages is informatively summarized in Table D.1.

Table D.1 – Persistence scope of SEI messages (informative)

SEI message	Persistence scope
Buffering period	The remainder of the bitstream
Picture timing	The access unit containing the SEI message
Pan-scan rectangle	Specified by the syntax of the SEI message
Filler payload	The access unit containing the SEI message
User data registered by Rec. ITU-T T.35	Unspecified
User data unregistered	Unspecified
Recovery point	Specified by the syntax of the SEI message
Scene information	The access unit containing the SEI message and up to but not including the next access unit, in decoding order, that contains a scene information SEI message or starts a new CLVS
Picture snapshot	The access unit containing the SEI message
Progressive refinement segment start	Specified by the syntax of the SEI message
Progressive refinement segment end	The access unit containing the SEI message
Film grain characteristics	Specified by the syntax of the SEI message
Post-filter hint	The access unit containing the SEI message
Tone mapping information	Specified by the syntax of the SEI message
Frame packing arrangement	Specified by the syntax of the SEI message
Display orientation	Specified by the syntax of the SEI message
Green metadata	Specified by the syntax of the SEI message
Structure of pictures information	The set of pictures in the coded layer-wise video sequence (CLVS) that correspond to entries listed in the SEI message
Decoded picture hash	The access unit containing the SEI message
Active parameter sets	The CVS containing the SEI message
Decoding unit information	The decoding unit containing the SEI message
Temporal sub-layer zero index	The access unit containing the SEI message
Scalable nesting	Depending on the scalable-nested SEI messages. Each scalable-nested SEI message has the same persistence scope as if the SEI message was not scalable-nested

Table D.1 – Persistence scope of SEI messages (informative)

SEI message	Persistence scope
Region refresh information	The set of VCL NAL units within the access unit starting from the VCL NAL unit following the SEI message up to but not including the VCL NAL unit following the next SEI NAL unit containing a region refresh information SEI message (if any)
No display	The access unit containing the SEI message
Time code	The access unit containing the SEI message
Mastering display colour volume	The CLVS containing the SEI message
Segmented rectangular frame packing arrangement	Specified by the syntax of the SEI message
Temporal motion-constrained tile sets	The access unit containing the SEI message and up to but not including the next access unit, in decoding order, that contains an SEI message of the same type or starts a new CLVS
Chroma resampling filter hint	The CLVS containing the SEI message
Knee function information	Specified by the syntax of the SEI message
Colour remapping information	Specified by the syntax of the SEI message
Deinterlaced field identification	One or more pictures associated with the access unit containing the SEI message
Content light level information	The CLVS containing the SEI message
Dependent random access point indication	The access unit containing the SEI message
Coded region completion	The current slice segment associated with the SEI message
Alternative transfer characteristics	The CLVS containing the SEI message
Ambient viewing environment	The CLVS containing the SEI message
Content colour volume	Specified by the syntax of the SEI message
Equirectangular projection	Specified by the syntax of the SEI message
Cubemap projection	Specified by the syntax of the SEI message
Fisheye video information	Specified by the syntax of the SEI message
Sphere rotation	Specified by the syntax of the SEI message
Region-wise packing	Specified by the syntax of the SEI message
Omnidirectional viewport	Specified by the syntax of the SEI message
Regional nesting	Depending on the region-nested SEI messages; each region-nested SEI message has the same persistence scope as if the SEI message was non-region-nested
Motion-constrained tile sets extraction information sets	The access unit containing the SEI message and up to but not including the next access unit, in decoding order, that contains an SEI message of the same type or starts a new CLVS
Motion-constrained tile sets extraction information nesting	The access unit containing the SEI message
SEI manifest	The CVS containing the SEI message
SEI prefix indication	The CVS containing the SEI message
Annotated regions	Specified by the syntax of the SEI message
Shutter interval information	The CLVS containing the SEI message

The values of some SEI message syntax elements, including `pan_scan_rect_id`, `scene_id`, `second_scene_id`, `snapshot_id`, `progressive_refinement_id`, `tone_map_id`, `frame_packing_arrangement_id`, `mcts_id[i]`, `knee_function_id`, `colour_remap_id`, `ilcts_id[i]`, and `regional_nesting_id`, are split into two sets of value ranges, where the first set is specified as "may be used as determined by the application", and the second set is specified as "reserved for future use by ITU-T |

ISO/IEC". Applications should be cautious of potential "collisions" of the interpretation for values of these syntax elements belonging to the first set of value ranges. Since different applications might use these IDs having values in the first set of value ranges for different purposes, particular care should be exercised in the design of encoders that generate SEI messages with these IDs having values in the first set of value ranges, and in the design of decoders that interpret SEI messages with these IDs. This Specification does not define any management for these values. These IDs having values in the first set of value ranges might only be suitable for use in contexts in which "collisions" of usage (i.e., different definitions of the syntax and semantics of an SEI message with one of these IDs having the same value in the first set of value ranges) are unimportant, or not possible, or are managed – e.g., defined or managed in the controlling application or transport specification, or by controlling the environment in which bitstreams are distributed.

It is a requirement of bitstream conformance that when a prefix SEI message with payloadType equal to 17 (progressive refinement segment end) or 22 (post-filter hint) is present in an access unit, a suffix SEI message with the same value of payloadType shall not be present in the same access unit.

It is a requirement of bitstream conformance that the following restrictions apply on containing of SEI messages in SEI NAL units:

- An SEI NAL unit containing an active parameter sets SEI message shall contain only one active parameter sets SEI message and shall not contain any other SEI messages.
- When an SEI NAL unit contains a non-scalable-nested buffering period SEI message, a non-scalable-nested picture timing SEI message, or a non-scalable-nested decoding unit information SEI message, the SEI NAL unit shall not contain any other SEI message with payloadType not equal to 0 (buffering period), 1 (picture timing) or 130 (decoding unit information).
- When an SEI NAL unit contains a scalable-nested buffering period SEI message, a scalable-nested picture timing SEI message, or a scalable-nested decoding unit information SEI message, the SEI NAL unit shall not contain any other SEI message with payloadType not equal to 0 (buffering period), 1 (picture timing), 130 (decoding unit information) or 133 (scalable nesting).

Let prevVclNalUnitInAu of an SEI NAL unit or an SEI message be the preceding VCL NAL unit in decoding order, if any, in the same access unit, and nextVclNalUnitInAu of an SEI NAL unit or an SEI message be the next VCL NAL unit in decoding order, if any, in the same access unit.

It is a requirement of bitstream conformance that the following restrictions apply on decoding order of SEI messages:

- When an SEI NAL unit containing an active parameter sets SEI message is present in an access unit, it shall be the first SEI NAL unit that follows the prevVclNalUnitInAu of the SEI NAL unit and precedes the nextVclNalUnitInAu of the SEI NAL unit.
- When a non-scalable-nested buffering period SEI message is present in an access unit, it shall not follow any other SEI message that follows the prevVclNalUnitInAu of the buffering period SEI message and precedes the nextVclNalUnitInAu of the buffering period SEI message, other than an active parameter sets SEI message.
- When a non-scalable-nested picture timing SEI message is present in an access unit, it shall not follow any other SEI message that follows the prevVclNalUnitInAu of the picture timing SEI message and precedes the nextVclNalUnitInAu of the picture timing SEI message, other than an active parameter sets SEI message or a non-scalable-nested buffering period SEI message.
- When a non-scalable-nested decoding unit information SEI message is present in an access unit, it shall not follow any other SEI message in the same access unit that follows the prevVclNalUnitInAu of the decoding unit information SEI message and precedes the nextVclNalUnitInAu of the decoding unit information SEI message, other than an active parameter sets SEI message, a non-scalable-nested buffering period SEI message, or a non-scalable-nested picture timing SEI message.
- When a scalable-nested buffering period SEI message, a scalable-nested picture timing SEI message, or a scalable-nested decoding unit information SEI message is contained in a scalable nesting SEI message in an access unit, the scalable nesting SEI message shall not follow any other SEI message that follows the prevVclNalUnitInAu of the scalable nesting SEI message and precedes the nextVclNalUnitInAu of the scalable nesting SEI message, other than an active parameter sets SEI message, a non-scalable-nested buffering period SEI message, a non-scalable-nested picture timing SEI message, a non-scalable-nested decoding unit information SEI message, or another scalable nesting SEI message that contains a buffering period SEI message, a picture timing SEI message, or a decoding unit information SEI message.
- When payloadType is equal to 0 (buffering period), 1 (picture timing) or 130 (decoding unit information) for an SEI message, scalable-nested or non-scalable-nested, within the access unit, the SEI NAL unit containing the SEI message shall precede all NAL units of any picture unit that has nuh_layer_id greater than highestAppLayerId, where highestAppLayerId is the greatest value of nuh_layer_id of all the layers in all the operation points that the SEI message applies to.

- When payloadType is equal to any value among VclAssociatedSeiList for an SEI message, scalable-nested or non-scalable-nested, within the access unit, the SEI NAL unit containing the SEI message shall precede all NAL units of any picture unit that has nuh_layer_id greater than highestAppLayerId, where highestAppLayerId is the greatest value of nuh_layer_id of all the layers that the SEI message applies to.

The following applies on the applicable operation points or layers of SEI messages:

- For a non-scalable-nested SEI message, when payloadType is equal to 0 (buffering period) or 130 (decoding unit information), the non-scalable-nested SEI message applies to the operation point that has OpTid equal to the greatest value of nuh_temporal_id_plus1 among all VCL NAL units in the bitstream, has OpLayerIdList containing all values of nuh_layer_id in all VCL units in the bitstream, and has only the base layer as the output layer.
- An SEI message that is directly contained in a scalable nesting SEI message within an SEI NAL unit with nuh_layer_id equal to 0 and has payloadType is equal to 0 (buffering period), 1 (picture timing), or 130 (decoding unit information) applies as specified in Annex C to the layer set as indicated by the scalable nesting SEI message.
- For a non-scalable-nested SEI message, when payloadType is equal to 1 (picture timing), the frame field information carried in the syntax elements pic_struct, source_scan_type and duplicate_flag, when present, in the non-scalable-nested picture timing SEI message applies to the base layer only, while the picture timing information carried in other syntax elements, when present, in the non-scalable-nested picture timing SEI message applies to the operation point that has OpTid equal to the greatest value of nuh_temporal_id_plus1 among all VCL NAL units in the bitstream, has OpLayerIdList containing all values of nuh_layer_id in all VCL units in the bitstream, and has only the base layer as the output layer.
- For a non-scalable-nested SEI message, when payloadType is equal to any value among VclAssociatedSeiList, the non-scalable-nested SEI message applies to the layer for which the VCL NAL units have nuh_layer_id equal to the nuh_layer_id of the SEI NAL unit containing the SEI message.
- An active parameter sets SEI message, which cannot be scalable-nested, applies to all layers in the bitstream.

It is a requirement of bitstream conformance that the following restrictions apply on the values of nuh_layer_id and TemporalId of SEI NAL units:

- When a non-scalable-nested SEI message has payloadType equal to any value among VclAssociatedSeiList, the SEI NAL unit containing the non-scalable-nested SEI message shall have TemporalId equal to the TemporalId of the access unit containing the SEI NAL unit.
- When a non-scalable-nested SEI message has payloadType equal to 0, 1, 129 or 130, the SEI NAL unit containing the non-scalable-nested SEI message shall have nuh_layer_id equal to 0.
- When a non-scalable-nested SEI message has payloadType equal to any value among VclAssociatedSeiList, the SEI NAL unit containing the non-scalable-nested SEI message shall have nuh_layer_id and nuh_temporal_id_plus1 equal to the values of nuh_layer_id and nuh_temporal_id_plus1, respectively, of the VCL NAL unit associated with the SEI NAL unit.

NOTE 4 – For an SEI NAL unit containing a scalable nesting SEI message, the values of TemporalId and nuh_layer_id should be set equal to the lowest value of TemporalId and nuh_layer_id, respectively, of all the sub-layers or operation points the scalable-nested SEI messages apply to unless specified otherwise.

It is a requirement of bitstream conformance that the following restrictions apply on the presence of SEI messages between two VCL NAL units of a picture:

- When there is a prefix SEI message that has payloadType equal to any value among SingleLayerSeiList not equal to 134 (the region refresh information SEI message) or 146 (the coded region completion SEI message), and applies to a picture of a layer layerA present between two VCL NAL units of the picture in decoding order, there shall be a prefix SEI message that is of the same type and applies to the layer layerA in the same access unit preceding the first VCL NAL unit of the picture.
- When there is a suffix SEI message that has payloadType equal to 3 (filler payload), 17 (progressive refinement segment end), 22 (post filter hint) or 132 (decoded picture hash) and applies to a picture of a layer layerA present between two VCL NAL units of the picture in decoding order, there shall be a suffix SEI message that is of the same type and applies to the layer layerA present in the same access unit succeeding the last VCL NAL unit of the picture.

It is a requirement of bitstream conformance that the following restrictions apply on repetition of SEI messages:

- For each of the payloadType values included in PicUnitRepConSeiList, there shall be less than or equal to 8 identical sei_payload() syntax structures within a picture unit.
- There shall be less than or equal to 8 identical sei_payload() syntax structures with payloadType equal to 130 within a decoding unit.

- The number of identical sei_payload() syntax structures with payloadType equal to 134 in a picture unit shall be less than or equal to the number of slice segments in the picture unit.

In the following subclauses of this annex, the following applies:

- The current SEI message refers to the particular SEI message.
- The current access unit refers to the access unit containing the current SEI message.

In the following subclauses of this annex, when a particular SEI message applies to a set of one or more layers (instead of a set of operation points), i.e., when the payloadType value is not equal to one of 0 (buffering period), 1 (picture timing) and 130 (decoding unit information), the following applies:

- The semantics apply independently to each particular layer with nuh_layer_id equal to targetLayerId of the layers to which the particular SEI message applies.
- The current layer refers to the layer with nuh_layer_id equal to targetLayerId.
- The current picture or the current decoded picture refers to the picture with nuh_layer_id equal to targetLayerId (i.e., in the current layer) in the current access unit.

In the following subclauses of this annex, when a particular SEI message applies to a set of one or more operation points (instead of a set of one or more layers), i.e., when the payloadType value is equal to 0 (buffering period), 1 (picture timing) or 130 (decoding unit information), the following applies:

- When the particular SEI message applies to an operation point that does not include the base layer (i.e., when the SEI message is contained in an SEI NAL unit with nuh_layer_id greater than 0), decoders conforming to a profile specified in Annex A and not supporting the INBLD capability specified in Annex F shall ignore that particular SEI message.
- The semantics apply independently to each particular operation point of the set of operation points to which the particular SEI message applies.
- The current operation point refers to the particular operation point.
- The terms "access unit" and "CVS" apply to the bitstream BitstreamToDecode that is the sub-bitstream of the particular operation point.

D.3.2 Buffering period SEI message semantics

A buffering period SEI message provides initial CPB removal delay and initial CPB removal delay offset information for initialization of the HRD at the position of the associated access unit in decoding order.

When the buffering period SEI message is non-scalable-nested or is directly contained in a scalable nesting SEI message within an SEI NAL unit with nuh_layer_id equal to 0 and the current access unit is a CRA or BLA access unit, let skippedPictureList be the list of skipped leading pictures consisting of the RASL pictures associated with the CRA or BLA access unit with which the buffering period SEI message is associated.

When the buffering period SEI message is non-scalable-nested or is directly contained in a scalable nesting SEI message within an SEI NAL unit with nuh_layer_id equal to 0, a picture is said to be a notDiscardablePic picture when the picture has TemporalId equal to 0 and is not a RASL, RADL or SLNR picture.

When the buffering period SEI message is non-scalable-nested or is directly contained in a scalable nesting SEI message within an SEI NAL unit with nuh_layer_id equal to 0, the following applies for the buffering period SEI message syntax and semantics:

- The syntax elements initial_cpb_removal_delay_length_minus1, au_cpb_removal_delay_length_minus1, dpb_output_delay_length_minus1, and sub_pic_hrd_params_present_flag and the variables NalHrdBpPresentFlag and VclHrdBpPresentFlag are found in or derived from syntax elements found in the hrd_parameters() syntax structure that is applicable to at least one of the operation points to which the buffering period SEI message applies.
- The variables CpbSize[i], BitRate[i] and CpbCnt are derived from syntax elements found in the sub_layer_hrd_parameters() syntax structure that is applicable to at least one of the operation points to which the buffering period SEI message applies.
- Any two operation points that the buffering period SEI message applies to having different OpTid values tIdA and tIdB indicate that the values of cpb_cnt_minus1[tIdA] and cpb_cnt_minus1[tIdB] coded in the hrd_parameters() syntax structure(s) applicable to the two operation points are identical.
- Any two operation points that the buffering period SEI message applies to having different OpLayerIdList values layerIdListA and layerIdListB indicate that the values of nal_hrd_parameters_present_flag and vcl_hrd_parameters_present_flag, respectively, for the two hrd_parameters() syntax structures applicable to the two operation points are identical.

- The bitstream (or a part thereof) refers to the bitstream subset (or a part thereof) associated with any of the operation points to which the buffering period SEI message applies.

The presence of buffering period SEI messages for an operation point including the base layer is specified as follows:

- If `NalHrdBpPresentFlag` is equal to 1 or `VclHrdBpPresentFlag` is equal to 1, the following applies for each access unit in the CVS:
 - If the access unit is an IRAP access unit, a buffering period SEI message applicable to the operation point shall be associated with the access unit.
 - Otherwise, if the access unit contains a `notDiscardablePic`, a buffering period SEI message applicable to the operation point may or may not be associated with the access unit.
 - Otherwise, the access unit shall not be associated with a buffering period SEI message applicable to the operation point.
- Otherwise (`NalHrdBpPresentFlag` and `VclHrdBpPresentFlag` are both equal to 0), no access unit in the CVS shall be associated with a buffering period SEI message applicable to the operation point.

NOTE 1 – For some applications, frequent presence of buffering period SEI messages may be desirable (e.g., for random access at an IRAP picture or a non-IRAP picture or for bitstream splicing).

bp_seq_parameter_set_id indicates and shall be equal to the `sps_seq_parameter_set_id` for the SPS that is active for the coded picture associated with the buffering period SEI message. The value of `bp_seq_parameter_set_id` shall be equal to the value of `pps_seq_parameter_set_id` in the PPS referenced by the `slice_pic_parameter_set_id` of the slice segment headers of the coded picture associated with the buffering period SEI message. The value of `bp_seq_parameter_set_id` shall be in the range of 0 to 15, inclusive.

irap_cpb_params_present_flag equal to 1 specifies the presence of the `nal_initial_alt_cpb_removal_delay[i]` and `nal_initial_alt_cpb_removal_offset[i]` or `vcl_initial_alt_cpb_removal_delay[i]` and `vcl_initial_alt_cpb_removal_offset[i]` syntax elements. When not present, the value of `irap_cpb_params_present_flag` is inferred to be equal to 0. When the associated picture is neither a CRA picture nor a BLA picture, the value of `irap_cpb_params_present_flag` shall be equal to 0.

NOTE 2 – The values of `sub_pic_hrd_params_present_flag` and `irap_cpb_params_present_flag` cannot be both equal to 1.

cpb_delay_offset specifies an offset to be used in the derivation of the nominal CPB removal times of access units following, in decoding order, the CRA or BLA access unit associated with the buffering period SEI message when no picture in `skippedPictureList` is present. The syntax element has a length in bits given by `au_cpb_removal_delay_length_minus1 + 1`. When not present, the value of `cpb_delay_offset` is inferred to be equal to 0.

dpb_delay_offset specifies an offset to be used in the derivation of the DPB output times of the CRA or BLA access unit associated with the buffering period SEI message when no picture in `skippedPictureList` is present. The syntax element has a length in bits given by `dpb_output_delay_length_minus1 + 1`. When not present, the value of `dpb_delay_offset` is inferred to be equal to 0.

When the current picture is not the first picture in the bitstream in decoding order, let `prevNonDiscardablePic` be the preceding picture in decoding order with `TemporalId` equal to 0 that is not a RASL, RADL or SLNR picture.

concatenation_flag indicates, when the current picture is not the first picture in the bitstream in decoding order, whether the nominal CPB removal time of the current picture is determined relative to the nominal CPB removal time of the preceding picture with a buffering period SEI message or relative to the nominal CPB removal time of the picture `prevNonDiscardablePic`.

au_cpb_removal_delay_delta_minus1 plus 1, when the current picture is not the first picture in the bitstream in decoding order, specifies a CPB removal delay increment value relative to the nominal CPB removal time of the picture `prevNonDiscardablePic`. This syntax element has a length in bits given by `au_cpb_removal_delay_length_minus1 + 1`.

When the current picture contains a buffering period SEI message and `concatenation_flag` is equal to 0 and the current picture is not the first picture in the bitstream in decoding order, it is a requirement of bitstream conformance that the following constraint applies:

- If the picture `prevNonDiscardablePic` is not associated with a buffering period SEI message, the `au_cpb_removal_delay_minus1` of the current picture shall be equal to the `au_cpb_removal_delay_minus1` of `prevNonDiscardablePic` plus `au_cpb_removal_delay_delta_minus1 + 1`.
- Otherwise, `au_cpb_removal_delay_minus1` shall be equal to `au_cpb_removal_delay_delta_minus1`.

NOTE 3 – When the current picture contains a buffering period SEI message and `concatenation_flag` is equal to 1, the `au_cpb_removal_delay_minus1` for the current picture is not used. The above-specified constraint can, under some circumstances, make it possible to splice bitstreams (that use suitably-designed referencing structures) by simply changing the value of `concatenation_flag` from 0 to 1 in the buffering period SEI message for an IRAP picture at the splicing point. When

concatenation_flag is equal to 0, the above-specified constraint enables the decoder to check whether the constraint is satisfied as a way to detect the loss of the picture prevNonDiscardablePic.

nal_initial_cpb_removal_delay[i] and **nal_initial_alt_cpb_removal_delay[i]** specify the default and the alternative initial CPB removal delays, respectively, for the i-th CPB when the NAL HRD parameters are in use. The syntax elements have a length in bits given by **initial_cpb_removal_delay_length_minus1 + 1**, and are in units of a 90 kHz clock. The values of the syntax elements shall not be equal to 0 and shall be less than or equal to $90\,000 * (\text{CpbSize}[i] \div \text{BitRate}[i])$, the time-equivalent of the CPB size in 90 kHz clock units.

nal_initial_cpb_removal_offset[i] and **nal_initial_alt_cpb_removal_offset[i]** specify the default and the alternative initial CPB removal offsets, respectively, for the i-th CPB when the NAL HRD parameters are in use. The syntax elements have a length in bits given by **initial_cpb_removal_delay_length_minus1 + 1** and are in units of a 90 kHz clock.

Over the entire CVS, the sum of **nal_initial_cpb_removal_delay[i]** and **nal_initial_cpb_removal_offset[i]** shall be constant for each value of i, and the sum of **nal_initial_alt_cpb_removal_delay[i]** and **nal_initial_alt_cpb_removal_offset[i]** shall be constant for each value of i.

vcl_initial_cpb_removal_delay[i] and **vcl_initial_alt_cpb_removal_delay[i]** specify the default and the alternative initial CPB removal delays, respectively, for the i-th CPB when the VCL HRD parameters are in use. The syntax elements have a length in bits given by **initial_cpb_removal_delay_length_minus1 + 1**, and are in units of a 90 kHz clock. The values of the syntax elements shall not be equal to 0 and shall be less than or equal to $90\,000 * (\text{CpbSize}[i] \div \text{BitRate}[i])$, the time-equivalent of the CPB size in 90 kHz clock units.

vcl_initial_cpb_removal_offset[i] and **vcl_initial_alt_cpb_removal_offset[i]** specify the default and the alternative initial CPB removal offsets, respectively, for the i-th CPB when the VCL HRD parameters are in use. The syntax elements have a length in bits given by **initial_cpb_removal_delay_length_minus1 + 1** and are in units of a 90 kHz clock.

Over the entire CVS, the sum of **vcl_initial_cpb_removal_delay[i]** and **vcl_initial_cpb_removal_offset[i]** shall be constant for each value of i, and the sum of **vcl_initial_alt_cpb_removal_delay[i]** and **vcl_initial_alt_cpb_removal_offset[i]** shall be constant for each value of i.

NOTE 4 – Encoders are recommended not to include **irap_cpb_params_present_flag** equal to 1 in buffering period SEI messages associated with a CRA or BLA picture for which at least one of its associated RASL pictures follows one or more of its associated RADL pictures in decoding order.

use_alt_cpb_params_flag may be used to derive the value of **UseAltCpbParamsFlag**. When **irap_cpb_params_present_flag** is equal to 0, **use_alt_cpb_params_flag** shall not be equal to 1. When **use_alt_cpb_params_flag** is not present, it is inferred to be equal to 0.

NOTE 5 – The syntax element **use_alt_cpb_params_flag** may be present in the payload extension of the buffering period SEI message. Decoders conforming to profiles specified in Annex A may ignore this syntax element.

It is a requirement of bitstream conformance that when **use_alt_cpb_params_flag** is present in the buffering period SEI message, the return value of the **more_data_in_payload()** function in the **sei_payload()** syntax structure containing the buffering period SEI message shall be equal to 1.

D.3.3 Picture timing SEI message semantics

The picture timing SEI message provides CPB removal delay and DPB output delay information for the access unit associated with the SEI message.

When the buffering period SEI message is non-scalable-nested or is directly contained in a scalable nesting SEI message within an SEI NAL unit with **nuh_layer_id** equal to 0, the following applies for the picture timing SEI message syntax and semantics:

- The syntax elements **sub_pic_hrd_params_present_flag**, **sub_pic_cpb_params_in_pic_timing_sei_flag**, **au_cpb_removal_delay_length_minus1**, **dpb_output_delay_length_minus1**, **dpb_output_delay_du_length_minus1**, **du_cpb_removal_delay_increment_length_minus1** and the variable **CpbDpbDelaysPresentFlag** are found in or derived from syntax elements found in the **hrd_parameters()** syntax structure that is applicable to at least one of the operation points to which the picture timing SEI message applies.
- The bitstream (or a part thereof) refers to the bitstream subset (or a part thereof) associated with any of the operation points to which the picture timing SEI message applies.

NOTE 1 – The syntax of the picture timing SEI message is dependent on the content of the **hrd_parameters()** syntax structures applicable to the operation points to which the picture timing SEI message applies. These **hrd_parameters()** syntax structures are in either or both of the VPS and the SPS that are active for the coded picture associated with the picture timing SEI message. When the picture timing SEI message is associated with an IRAP access unit with **NoRasOutputFlag** equal to 1, unless it is preceded by an active parameter sets SEI message within the same access unit, the activation of the VPS and the SPS (and, for IRAP pictures with **NoRasOutputFlag** equal to 1 that are not the first picture in the bitstream in decoding order, the determination that the coded picture is an IRAP access unit with **NoRasOutputFlag** equal to 1) does not occur

until the decoding of the first coded slice segment NAL unit of the coded picture. Since the coded slice segment NAL unit of the coded picture follows the picture timing SEI message in NAL unit order, there may be cases in which it is necessary for a decoder to store the RBSP containing the picture timing SEI message until determining the active VPS and the active SPS for the coded picture, and then perform the parsing of the picture timing SEI message.

The presence of picture timing SEI messages for an operation point including the base layer is specified as follows:

- If `frame_field_info_present_flag` is equal to 1 or `CpbDpbDelaysPresentFlag` is equal to 1, a picture timing SEI message applicable to the operation point shall be associated with every access unit in the CVS.
- Otherwise, in the CVS there shall be no access unit that is associated with a picture timing SEI message applicable to the operation point.

When the picture timing SEI message is not scalable-nested or is directly contained in a scalable nesting SEI message within an SEI NAL unit with `nuh_layer_id` equal to 0, the semantics of `pic_struct`, `source_scan_type` and `duplicate_flag` apply to the picture with `nuh_layer_id` equal to 0 and are specified in the following paragraphs.

NOTE 2 – When the picture timing SEI message is directly contained in a scalable nesting SEI message within an SEI NAL unit with `nuh_layer_id` greater than 0 or is contained in a bitstream partition nesting SEI message specified in Annex F, the semantics of `pic_struct`, `source_scan_type` and `duplicate_flag` are specified in Annex F. The frame-field information SEI message specified in Annex F can be used to indicate `pic_struct`, `source_scan_type` and `duplicate_flag` for non-base layers.

pic_struct indicates whether a picture should be displayed as a frame or as one or more fields and, for the display of frames when `fixed_pic_rate_within_cvs_flag` is equal to 1, may indicate a frame doubling or tripling repetition period for displays that use a fixed frame refresh interval equal to `DpbOutputElementalInterval[n]` as given by Equation E-76. The interpretation of `pic_struct` is specified in Table D.2. Values of `pic_struct` that are not listed in Table D.2 are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore reserved values of `pic_struct`.

When present, it is a requirement of bitstream conformance that the value of `pic_struct` shall be constrained such that exactly one of the following conditions is true:

- The value of `pic_struct` is equal to 0, 7 or 8 for all pictures in the CVS.
- The value of `pic_struct` is equal to 1, 2, 9, 10, 11 or 12 for all pictures in the CVS.
- The value of `pic_struct` is equal to 3, 4, 5 or 6 for all pictures in the CVS.

When `fixed_pic_rate_within_cvs_flag` is equal to 1, frame doubling is indicated by `pic_struct` equal to 7, which indicates that the frame should be displayed two times consecutively on displays with a frame refresh interval equal to `DpbOutputElementalInterval[n]` as given by Equation E-76, and frame tripling is indicated by `pic_struct` equal to 8, which indicates that the frame should be displayed three times consecutively on displays with a frame refresh interval equal to `DpbOutputElementalInterval[n]` as given by Equation E-76.

NOTE 3 – Frame doubling can be used to facilitate the display, for example, of 25 Hz progressive-scan video on a 50 Hz progressive-scan display or 30 Hz progressive-scan video on a 60 Hz progressive-scan display. Using frame doubling and frame tripling in alternating combination on every other frame can be used to facilitate the display of 24 Hz progressive-scan video on a 60 Hz progressive-scan display.

The nominal vertical and horizontal sampling locations of samples in top and bottom fields for 4:2:0, 4:2:2 and 4:4:4 chroma formats are shown in Figure D.1, Figure D.2 and Figure D.3, respectively.

Association indicators for fields (`pic_struct` equal to 9 through 12) provide hints to associate fields of complementary parity together as frames. The parity of a field can be top or bottom, and the parity of two fields is considered complementary when the parity of one field is top and the parity of the other field is bottom.

When `frame_field_info_present_flag` is equal to 1, it is a requirement of bitstream conformance that the constraints specified in the third column of Table D.2 shall apply.

NOTE 4 – When `frame_field_info_present_flag` is equal to 0, then in many cases default values may be inferred or indicated by other means. In the absence of other indications of the intended display type of a picture, the decoder should infer the value of `pic_struct` as equal to 0 when `frame_field_info_present_flag` is equal to 0.

source_scan_type equal to 1 indicates that the source scan type of the associated picture should be interpreted as progressive. `source_scan_type` equal to 0 indicates that the source scan type of the associated picture should be interpreted as interlaced. `source_scan_type` equal to 2 indicates that the source scan type of the associated picture is unknown or unspecified. `source_scan_type` equal to 3 is reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders conforming to this version of this Specification shall interpret the value 3 for `source_scan_type` as equivalent to the value 2.

The following applies to the semantics of `source_scan_type`:

- If `general_progressive_source_flag` is equal to 0 and `general_interlaced_source_flag` is equal to 1, the value of `source_scan_type` shall be equal to 0 when present, and should be inferred to be equal to 0 when not present.
- Otherwise, if `general_progressive_source_flag` is equal to 1 and `general_interlaced_source_flag` is equal to 0, the value of `source_scan_type` shall be equal to 1 when present and should be inferred to be equal to 1 when not present.
- Otherwise, when `general_progressive_source_flag` is equal to 0 and `general_interlaced_source_flag` is equal to 0, the value of `source_scan_type` shall be equal to 2 when present and should be inferred to be equal to 2 when not present.

duplicate_flag equal to 1 indicates that the current picture is indicated to be a duplicate of a previous picture in output order. **duplicate_flag** equal to 0 indicates that the current picture is not indicated to be a duplicate of a previous picture in output order.

NOTE 5 – The `duplicate_flag` should be used to mark coded pictures known to have originated from a repetition process such as 3:2 pull-down or other such duplication and picture rate interpolation methods. This flag would commonly be used when a video feed is encoded as a field sequence in a "transport pass-through" fashion, with known duplicate pictures tagged by setting `duplicate_flag` equal to 1.

NOTE 6 – When `field_seq_flag` is equal to 1 and `duplicate_flag` is equal to 1, this should be interpreted as an indication that the access unit contains a duplicated field of the previous field in output order with the same parity as the current field unless a pairing is otherwise indicated by the use of a `pic_struct` value in the range of 9 to 12, inclusive.

Table D.2 – Interpretation of `pic_struct`

Value	Indicated display of picture	Restrictions
0	(progressive) Frame	<code>field_seq_flag</code> shall be equal to 0
1	Top field	<code>field_seq_flag</code> shall be equal to 1
2	Bottom field	<code>field_seq_flag</code> shall be equal to 1
3	Top field, bottom field, in that order	<code>field_seq_flag</code> shall be equal to 0
4	Bottom field, top field, in that order	<code>field_seq_flag</code> shall be equal to 0
5	Top field, bottom field, top field repeated, in that order	<code>field_seq_flag</code> shall be equal to 0
6	Bottom field, top field, bottom field repeated, in that order	<code>field_seq_flag</code> shall be equal to 0
7	Frame doubling	<code>field_seq_flag</code> shall be equal to 0 <code>fixed_pic_rate_within_cvs_flag</code> shall be equal to 1
8	Frame tripling	<code>field_seq_flag</code> shall be equal to 0 <code>fixed_pic_rate_within_cvs_flag</code> shall be equal to 1
9	Top field paired with previous bottom field in output order	<code>field_seq_flag</code> shall be equal to 1
10	Bottom field paired with previous top field in output order	<code>field_seq_flag</code> shall be equal to 1
11	Top field paired with next bottom field in output order	<code>field_seq_flag</code> shall be equal to 1
12	Bottom field paired with next top field in output order	<code>field_seq_flag</code> shall be equal to 1

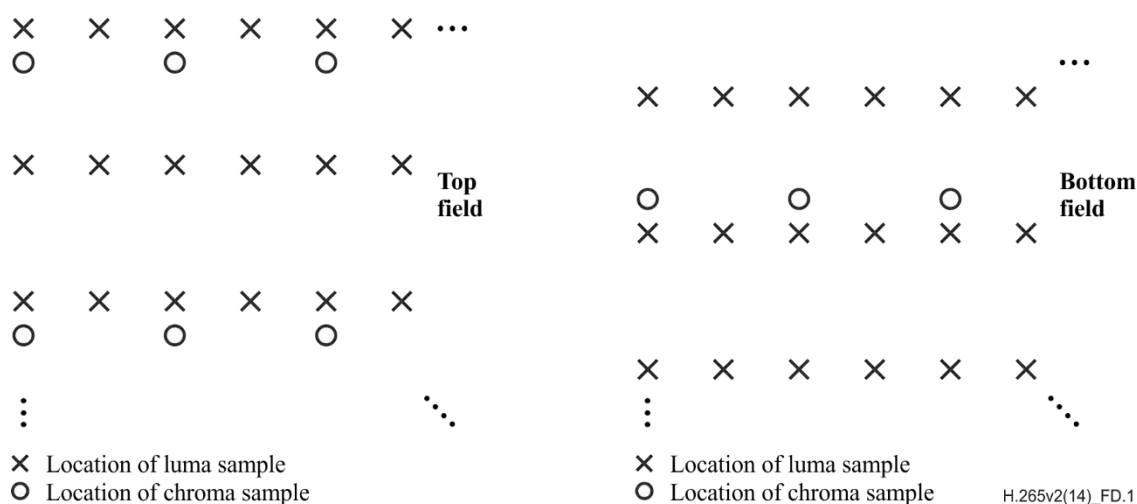


Figure D.1 – Nominal vertical and horizontal sampling locations of 4:2:0 samples in top and bottom fields

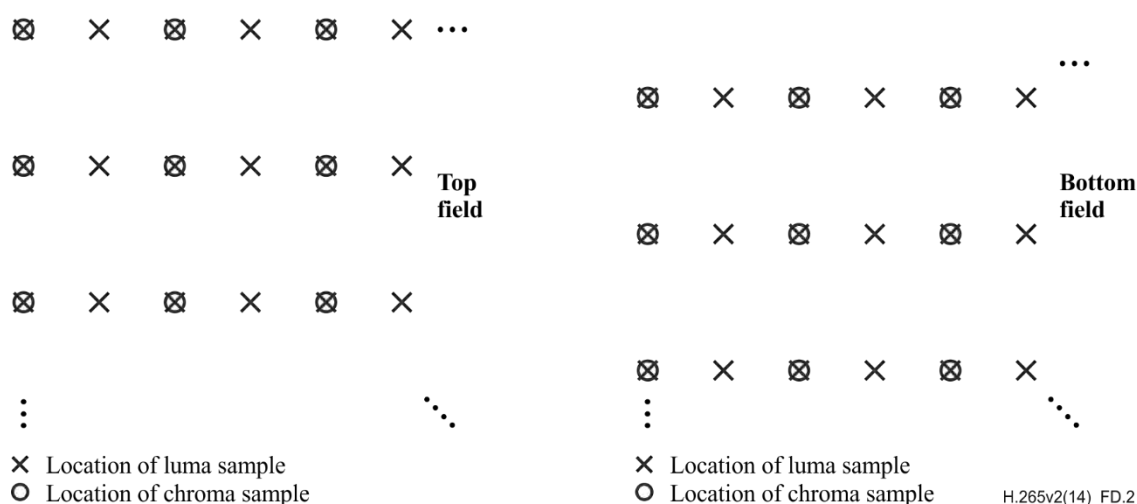


Figure D.2 – Nominal vertical and horizontal sampling locations of 4:2:2 samples in top and bottom fields

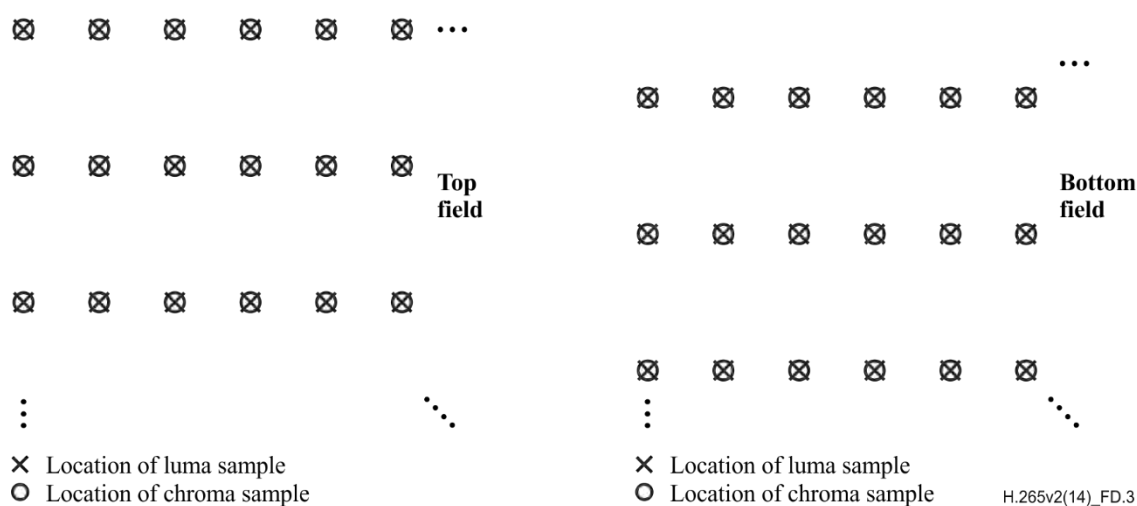


Figure D.3 – Nominal vertical and horizontal sampling locations of 4:4:4 samples in top and bottom fields

au_cpb_removal_delay_minus1 plus 1 is used to calculate the number of clock ticks between the nominal CPB removal times of the access unit associated with the picture timing SEI message and the preceding access unit in decoding order

that contained a buffering period SEI message. This value is also used to calculate an earliest possible time of arrival of access unit data into the CPB for the HSS. The syntax element is a fixed length code whose length in bits is given by $\text{au_cpb_removal_delay_length_minus1} + 1$.

NOTE 7 – The value of $\text{au_cpb_removal_delay_length_minus1}$ that determines the length (in bits) of the syntax element $\text{au_cpb_removal_delay_minus1}$ is the value of $\text{au_cpb_removal_delay_length_minus1}$ coded in the VPS or the SPS that is active for the coded picture associated with the picture timing SEI message, although the preceding access unit containing a buffering period SEI message may be an access unit of a different CVS.

The variable BpResetFlag of the current picture is derived as follows:

- If the current picture is associated with a buffering period SEI message that is applicable to at least one of the operation points to which the picture timing SEI message applies, BpResetFlag is set equal to 1.
- Otherwise, BpResetFlag is set equal to 0.

The variables $\text{AuCpbRemovalDelayMsb}$ and $\text{AuCpbRemovalDelayVal}$ of the current picture are derived as follows:

- If the current access unit is the access unit that initializes the HRD, $\text{AuCpbRemovalDelayMsb}$ and $\text{AuCpbRemovalDelayVal}$ are both set equal to 0.
- Otherwise, let the picture $\text{prevNonDiscardablePic}$ be the previous picture in decoding order that has TemporalId equal to 0 that is not a RASL, RADL or SLNR picture, let $\text{prevAuCpbRemovalDelayMinus1}$, $\text{prevAuCpbRemovalDelayMsb}$ and prevBpResetFlag be set equal to the values of $\text{au_cpb_removal_delay_minus1}$, $\text{AuCpbRemovalDelayMsb}$ and BpResetFlag , respectively, for the picture $\text{prevNonDiscardablePic}$, and the following applies:
 - $\text{AuCpbRemovalDelayMsb}$ is derived as follows:

```

if( prevBpResetFlag )
    AuCpbRemovalDelayMsb = 0
else if( au_cpb_removal_delay_minus1 <= prevAuCpbRemovalDelayMinus1 )
    AuCpbRemovalDelayMsb = prevAuCpbRemovalDelayMsb + 2au_cpb_removal_delay_length_minus1 + 1
    (D-1)
else
    AuCpbRemovalDelayMsb = prevAuCpbRemovalDelayMsb

```

- $\text{AuCpbRemovalDelayVal}$ is derived as follows:

$$\text{AuCpbRemovalDelayVal} = \text{AuCpbRemovalDelayMsb} + \text{au_cpb_removal_delay_minus1} + 1$$

(D-2)

The value of $\text{AuCpbRemovalDelayVal}$ shall be in the range of 1 to 2^{32} , inclusive.

pic_dpb_output_delay is used to compute the DPB output time of the picture when SubPicHrdFlag is equal to 0. It specifies how many clock ticks to wait after removal of the last decoding unit in an access unit from the CPB before the decoded picture is output from the DPB.

NOTE 8 – A picture is not removed from the DPB at its output time when it is still marked as "used for short-term reference" or "used for long-term reference".

The length of the syntax element $\text{pic_dpb_output_delay}$ is given in bits by $\text{dpb_output_delay_length_minus1} + 1$. When $\text{sps_max_dec_pic_buffering_minus1}[\text{minTid}]$ is equal to 0, where minTid is the minimum of the OpTid values of all operation points the picture timing SEI message applies to, $\text{pic_dpb_output_delay}$ shall be equal to 0.

The output time derived from the $\text{pic_dpb_output_delay}$ of any picture that is output from an output timing conforming decoder shall precede the output time derived from the $\text{pic_dpb_output_delay}$ of all pictures in any subsequent CVS in decoding order.

The picture output order established by the values of this syntax element shall be the same order as established by the values of PicOrderCntVal .

For pictures that are not output by the "bumping" process because they precede, in decoding order, an IRAP picture with NoRasOutputFlag equal to 1 that has $\text{no_output_of_prior_pics_flag}$ equal to 1 or inferred to be equal to 1, the output times derived from $\text{pic_dpb_output_delay}$ shall be increasing with increasing value of PicOrderCntVal relative to all pictures within the same CVS.

pic_dpb_output_du_delay is used to compute the DPB output time of the picture when SubPicHrdFlag is equal to 1. It specifies how many sub clock ticks to wait after removal of the last decoding unit in an access unit from the CPB before the decoded picture is output from the DPB.

The length of the syntax element `pic_dpb_output_du_delay` is given in bits by `dpb_output_delay_du_length_minus1 + 1`.

The output time derived from the `pic_dpb_output_du_delay` of any picture that is output from an output timing conforming decoder shall precede the output time derived from the `pic_dpb_output_du_delay` of all pictures in any subsequent CVS in decoding order.

The picture output order established by the values of this syntax element shall be the same order as established by the values of `PicOrderCntVal`.

For pictures that are not output by the "bumping" process because they precede, in decoding order, an IRAP picture with `NoRaslOutputFlag` equal to 1 that has `no_output_of_prior_pics_flag` equal to 1 or inferred to be equal to 1, the output times derived from `pic_dpb_output_du_delay` shall be increasing with increasing value of `PicOrderCntVal` relative to all pictures within the same CVS.

For any two pictures in the CVS, the difference between the output times of the two pictures when `SubPicHrdFlag` is equal to 1 shall be identical to the same difference when `SubPicHrdFlag` is equal to 0.

num_decoding_units_minus1 plus 1 specifies the number of decoding units in the access unit the picture timing SEI message is associated with. The value of `num_decoding_units_minus1` shall be in the range of 0 to `PicSizeInCtbsY - 1`, inclusive.

du_common_cpb_removal_delay_flag equal to 1 specifies that the syntax element `du_common_cpb_removal_delay_increment_minus1` is present. `du_common_cpb_removal_delay_flag` equal to 0 specifies that the syntax element `du_common_cpb_removal_delay_increment_minus1` is not present.

du_common_cpb_removal_delay_increment_minus1 plus 1 specifies the duration, in units of clock sub-ticks (see clause E.3.2), between the nominal CPB removal times of any two consecutive decoding units in decoding order in the access unit associated with the picture timing SEI message. This value is also used to calculate an earliest possible time of arrival of decoding unit data into the CPB for the HSS, as specified in Annex C or clause F.13. The syntax element is a fixed length code whose length in bits is given by `du_cpb_removal_delay_increment_length_minus1 + 1`.

num_nalus_in_du_minus1[i] plus 1 specifies the number of NAL units in the *i*-th decoding unit of the access unit the picture timing SEI message is associated with. The value of `num_nalus_in_du_minus1[i]` shall be in the range of 0 to `PicSizeInCtbsY - 1`, inclusive.

The first decoding unit of the access unit consists of the first `num_nalus_in_du_minus1[0] + 1` consecutive NAL units in decoding order in the access unit. The *i*-th (with *i* greater than 0) decoding unit of the access unit consists of the `num_nalus_in_du_minus1[i] + 1` consecutive NAL units immediately following the last NAL unit in the previous decoding unit of the access unit, in decoding order. There shall be at least one VCL NAL unit in each decoding unit. All non-VCL NAL units associated with a VCL NAL unit shall be included in the same decoding unit as the VCL NAL unit.

du_cpb_removal_delay_increment_minus1[i] plus 1 specifies the duration, in units of clock sub-ticks, between the nominal CPB removal times of the (*i* + 1)-th decoding unit and the *i*-th decoding unit, in decoding order, in the access unit associated with the picture timing SEI message. This value is also used to calculate an earliest possible time of arrival of decoding unit data into the CPB for the HSS, as specified in Annex C or clause F.13. The syntax element is a fixed length code whose length in bits is given by `du_cpb_removal_delay_increment_length_minus1 + 1`.

D.3.4 Pan-scan rectangle SEI message semantics

The pan-scan rectangle SEI message syntax elements specify the coordinates of one or more rectangles relative to the conformance cropping window specified by the active SPS. Each coordinate is specified in units of one-sixteenth luma sample spacing relative to the luma sampling grid.

pan_scan_rect_id contains an identifying number that may be used to identify the purpose of the one or more pan-scan rectangles (for example, to identify the one or more rectangles as the area to be shown on a particular display device or as the area that contains a particular actor in the scene). The value of `pan_scan_rect_id` shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of `pan_scan_rect_id` from 0 to 255, inclusive, and from 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of `pan_scan_rect_id` from 256 to 511, inclusive, and from 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of `pan_scan_rect_id` in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

pan_scan_rect_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous pan-scan rectangle SEI message in output order that applies to the current layer. `pan_scan_rect_cancel_flag` equal to 0 indicates that pan-scan rectangle information follows.

pan_scan_cnt_minus1 specifies the number of pan-scan rectangles that are specified by the SEI message. `pan_scan_cnt_minus1` shall be in the range of 0 to 2, inclusive.

`pan_scan_cnt_minus1` equal to 0 indicates that a single pan-scan rectangle is specified that applies to the decoded pictures that are within the persistence scope of the current SEI message. When `field_seq_flag` is equal to 1, `pan_scan_cnt_minus1` shall be equal to 0.

`pan_scan_cnt_minus1` equal to 1 indicates that two pan-scan rectangles are specified that apply to the decoded pictures that are within the persistence scope of the current SEI message and that are associated with picture timing SEI messages having `pic_struct` equal to 3 or 4. The first rectangle applies to the first field of a frame in output order and the second rectangle applies to the second field of a frame in output order, where the output order between two fields in one frame is as shown in Table D.2 for `pic_struct` equal to 3 or 4.

`pan_scan_cnt_minus1` equal to 2 indicates that three pan-scan rectangles are specified that apply to the decoded pictures that are within the persistence scope of the current SEI message and that are associated with picture timing SEI messages having `pic_struct` equal to 5 or 6. The first rectangle applies to the first field of the frame in output order, the second rectangle applies to the second field of the frame in output order, and the third rectangle applies to a repetition of the first field as a third field in output order, where the output order of fields in one frame is as shown in Table D.2 for `pic_struct` equal to 5 or 6.

`pan_scan_rect_left_offset[i]`, `pan_scan_rect_right_offset[i]`, `pan_scan_rect_top_offset[i]` and `pan_scan_rect_bottom_offset[i]`, specify, as signed integer quantities in units of one-sixteenth sample spacing relative to the luma sampling grid, the location of the *i*-th pan-scan rectangle. The values of each of these four syntax elements shall be in the range of $-2^{31} + 1$ to $2^{31} - 1$, inclusive.

The pan-scan rectangle is specified, in units of one-sixteenth sample spacing relative to a luma sampling grid, as the region with horizontal coordinates from $16 * \text{SubWidthC} * \text{conf_win_left_offset} + \text{pan_scan_rect_left_offset}[i]$ to $16 * (\text{CtbSizeY} * \text{PicWidthInCtbsY} - \text{SubWidthC} * \text{conf_win_right_offset}) + \text{pan_scan_rect_right_offset}[i] - 1$ and with vertical coordinates from $16 * \text{SubHeightC} * \text{conf_win_top_offset} + \text{pan_scan_rect_top_offset}[i]$ to $16 * (\text{CtbSizeY} * \text{PicHeightInCtbsY} - \text{SubHeightC} * \text{conf_win_bottom_offset}) + \text{pan_scan_rect_bottom_offset}[i] - 1$, inclusive. The value of $16 * \text{SubWidthC} * \text{conf_win_left_offset} + \text{pan_scan_rect_left_offset}[i]$ shall be less than or equal to $16 * (\text{CtbSizeY} * \text{PicWidthInCtbsY} - \text{SubWidthC} * \text{conf_win_right_offset}) + \text{pan_scan_rect_right_offset}[i] - 1$ and the value of $16 * \text{SubHeightC} * \text{conf_win_top_offset} + \text{pan_scan_rect_top_offset}[i]$ shall be less than or equal to $16 * (\text{CtbSizeY} * \text{PicHeightInCtbsY} - \text{SubHeightC} * \text{conf_win_bottom_offset}) + \text{pan_scan_rect_bottom_offset}[i] - 1$.

When the pan-scan rectangular area includes samples outside of the conformance cropping window, the region outside of the conformance cropping window may be filled with synthesized content (such as black video content or neutral grey video content) for display.

`pan_scan_rect_persistence_flag` specifies the persistence of the pan-scan rectangle SEI message for the current layer.

`pan_scan_rect_persistence_flag` equal to 0 specifies that the pan-scan rectangle information applies to the current decoded picture only.

Let `picA` be the current picture. `pan_scan_rect_persistence_flag` equal to 1 specifies that the pan-scan rectangle information persists for the current layer in output order until any of the following conditions are true:

- A new CLVS of the current layer begins.
- The bitstream ends.
- A picture `picB` in the current layer in an access unit containing a pan-scan rectangle SEI message with the same value of `pan_scan_rect_id` and applicable to the current layer is output for which `PicOrderCnt(picB)` is greater than `PicOrderCnt(picA)`, where `PicOrderCnt(picB)` and `PicOrderCnt(picA)` are the `PicOrderCntVal` values of `picB` and `picA`, respectively, immediately after the invocation of the decoding process for picture order count for `picB`.

D.3.5 Filler payload SEI message semantics

This SEI message contains a series of `payloadSize` bytes of value 0xFF, which can be discarded.

`ff_byte` shall be a byte having the value 0xFF.

D.3.6 User data registered by Recommendation ITU-T T.35 SEI message semantics

This SEI message contains user data registered as specified in Recommendation ITU-T T.35, the contents of which are not specified in this Specification.

`itu_t_t35_country_code` shall be a byte having a value specified as a country code by Annex A of Recommendation ITU-T T.35.

`itu_t_t35_country_code_extension_byte` shall be a byte having a value specified as a country code by Annex B of Recommendation ITU-T T.35.

`itu_t_t35_payload_byte` shall be a byte containing data registered as specified in Recommendation ITU-T T.35.

The ITU-T T.35 terminal provider code and terminal provider oriented code shall be contained in the first one or more bytes of the `itu_t_t35_payload_byte`, in the format specified by the Administration that issued the terminal provider code. Any remaining `itu_t_t35_payload_byte` data shall be data having syntax and semantics as specified by the entity identified by the ITU-T T.35 country code and terminal provider code.

D.3.7 User data unregistered SEI message semantics

This SEI message contains unregistered user data identified by a universal unique identifier (UUID), the contents of which are not specified in this Specification.

`uuid_iso_iec_11578` shall have a value specified as a UUID according to the procedures of Annex A of ISO/IEC 11578:1996.

`user_data_payload_byte` shall be a byte containing data having syntax and semantics as specified by the UUID generator.

D.3.8 Recovery point SEI message semantics

The recovery point SEI message assists a decoder in determining when the decoding process will produce acceptable pictures for display after the decoder initiates random access or after the encoder indicates a broken link in the CVS. When the decoding process is started with the access unit in decoding order associated with the recovery point SEI message, all decoded pictures at or subsequent to the recovery point in output order specified in this SEI message are indicated to be correct or approximately correct in content. Decoded pictures produced by random access at or before the picture associated with the recovery point SEI message need not be correct in content until the indicated recovery point, and the operation of the decoding process starting at the picture associated with the recovery point SEI message may contain references to pictures unavailable in the decoded picture buffer.

In addition, by use of the `broken_link_flag`, the recovery point SEI message can indicate to the decoder the location of some pictures in the bitstream that can result in serious visual artefacts when displayed, even when the decoding process was begun at the location of a previous IRAP access unit in decoding order.

NOTE 1 – The `broken_link_flag` can be used by encoders to indicate the location of a point after which the decoding process for the decoding of some pictures may cause references to pictures that, though available for use in the decoding process, are not the pictures that were used for reference when the bitstream was originally encoded (e.g., due to a splicing operation performed during the generation of the bitstream).

When random access is performed to start decoding from the access unit associated with the recovery point SEI message, the decoder operates as if the associated picture was the first picture in the bitstream in decoding order, and the variables `prevPicOrderCntLsb` and `prevPicOrderCntMsb` used in derivation of `PicOrderCntVal` are both set equal to 0.

NOTE 2 – When HRD information is present in the bitstream, a buffering period SEI message should be associated with the access unit associated with the recovery point SEI message in order to establish initialization of the HRD buffer model after a random access.

Any SPS or PPS RBSP that is referred to by a picture associated with a recovery point SEI message or by any picture following such a picture in decoding order shall be available to the decoding process prior to its activation, regardless of whether or not the decoding process is started at the beginning of the bitstream or with the access unit, in decoding order, that is associated with the recovery point SEI message.

`recovery_poc_cnt` specifies the recovery point of decoded pictures in output order. If there is a picture `picA` that follows the current picture (i.e., the picture associated with the current SEI message) in decoding order in the CVS and that has `PicOrderCntVal` equal to the `PicOrderCntVal` of the current picture plus the value of `recovery_poc_cnt`, the picture `picA` is referred to as the recovery point picture. Otherwise, the first picture in output order that has `PicOrderCntVal` greater than the `PicOrderCntVal` of the current picture plus the value of `recovery_poc_cnt` is referred to as the recovery point picture. The recovery point picture shall not precede the current picture in decoding order. All decoded pictures in output order are indicated to be correct or approximately correct in content starting at the output order position of the recovery point picture. The value of `recovery_poc_cnt` shall be in the range of $-\text{MaxPicOrderCntLsb} / 2$ to $\text{MaxPicOrderCntLsb} / 2 - 1$, inclusive.

`exact_match_flag` indicates whether decoded pictures at and subsequent to the specified recovery point in output order derived by starting the decoding process at the access unit associated with the recovery point SEI message will be an exact match to the pictures that would be produced by starting the decoding process at the location of a previous IRAP access unit, if any, in the bitstream. The value 0 indicates that the match may not be exact and the value 1 indicates that the match will be exact. When `exact_match_flag` is equal to 1, it is a requirement of bitstream conformance that the decoded pictures at and subsequent to the specified recovery point in output order derived by starting the decoding process at the access unit associated with the recovery point SEI message shall be an exact match to the pictures that would be produced by starting the decoding process at the location of a previous IRAP access unit, if any, in the bitstream.

NOTE 3 – When performing random access, decoders should infer all references to unavailable pictures as references to pictures containing only intra coding blocks and having sample values given by Y equal to $(1 \ll (\text{BitDepth}_Y - 1))$, Cb and Cr both equal to $(1 \ll (\text{BitDepth}_C - 1))$ (mid-level grey), regardless of the value of `exact_match_flag`.

When `exact_match_flag` is equal to 0, the quality of the approximation at the recovery point is chosen by the encoding process and is not specified in this Specification.

broken_link_flag indicates the presence or absence of a broken link in the NAL unit stream at the location of the recovery point SEI message and is assigned further semantics as follows:

- If `broken_link_flag` is equal to 1, pictures produced by starting the decoding process at the location of a previous IRAP access unit may contain undesirable visual artefacts to the extent that decoded pictures at and subsequent to the access unit associated with the recovery point SEI message in decoding order should not be displayed until the specified recovery point in output order.
- Otherwise (`broken_link_flag` is equal to 0), no indication is given regarding any potential presence of visual artefacts.

When the current picture is a BLA picture, the value of `broken_link_flag` shall be equal to 1.

Regardless of the value of the `broken_link_flag`, pictures subsequent to the specified recovery point in output order are specified to be correct or approximately correct in content.

D.3.9 Scene information SEI message semantics

A scene and a scene transition are herein defined as a set of consecutive pictures in output order.

NOTE 1 – Decoded pictures within one scene generally have similar content. The scene information SEI message is used to label pictures with scene identifiers and to indicate scene changes. The message specifies how the source pictures for the labelled pictures were created. The decoder may use the information to select an appropriate algorithm to conceal transmission errors. For example, a specific algorithm may be used to conceal transmission errors that occurred in pictures belonging to a gradual scene transition. Furthermore, the scene information SEI message may be used in a manner determined by the application, such as for indexing the scenes of a video sequence.

A scene information SEI message labels all pictures of the current layer, in decoding order, from the coded picture to which the SEI message is associated (inclusive) to the coded picture to which the next scene information SEI message applicable to the current layer (when present) in decoding order is associated (exclusive) or (otherwise) to the last picture in the CLVS (inclusive). These pictures are herein referred to as the target pictures.

NOTE 2 – The semantics of the scene information SEI message apply layer-wise. However, the scene information SEI message may be contained within a scalable nesting SEI message, which may help in reducing the number of scene information SEI messages, as scene changes and transitions apply across layers.

scene_info_present_flag equal to 0 indicates that the scene or scene transition to which the target pictures belong is unspecified. `scene_info_present_flag` equal to 1 indicates that the target pictures belong to the same scene or scene transition.

prev_scene_id_valid_flag equal to 0 specifies that the `scene_id` value of the picture preceding the first picture of the target pictures in output order is considered unspecified in the semantics of the syntax elements of this SEI message. `prev_scene_id_valid_flag` equal to 1 specifies that the `scene_id` value of the picture preceding the first picture of the target pictures in output order is specified by the previous scene information SEI message in decoding order. When the previous scene information SEI message applicable to the current layer is within the same CLVS as the current scene information SEI message, `prev_scene_id_valid_flag` shall be equal to 1.

NOTE 3 – When a current scene information SEI message is associated with the first picture, in decoding order, of a CLVS, `prev_scene_id_valid_flag` equal to 1 indicates that the `scene_id` values of the current scene information SEI message and the previous scene information SEI message applicable to the current layer in decoding order can be used to conclude whether their target pictures belong to the same scene or to different scenes.

NOTE 4 – When CVS B is concatenated to CVS A and CVS A represents a different scene than the scene CVS B represents, it should be noticed that the `scene_id` value specified for the last picture with a particular `nuh_layer_id` value of CVS A affects the semantics of the scene information SEI message associated with that particular `nuh_layer_id` value and the first picture, in decoding order, of CVS B, when the SEI message is present. Hence, as part of such a concatenation operation, the value of `prev_scene_id_valid_flag` should be set equal to 0 in the scene information SEI message associated with the first picture, in decoding order, of CVS B, when the SEI message is present.

scene_id identifies the scene to which the target pictures belong. When the value of `scene_transition_type` of the target pictures is less than 4, and the previous picture in output order is marked with a value of `scene_transition_type` less than 4, and the value of `scene_id` is the same as the value of `scene_id` of the previous picture in output order, this indicates that the source scene for the target pictures and the source scene for the previous picture (in output order) are considered by the encoder to have been the same scene. When the value of `scene_transition_type` of the target pictures is greater than 3, and the previous picture in output order is marked with a value of `scene_transition_type` less than 4, and the value of `scene_id` is the same as the value of `scene_id` of the previous picture in output order, this indicates that one of the source scenes for the target pictures and the source scene for the previous picture (in output order) are considered by the encoder to have been the same scene. When the value of `scene_id` is not equal to the value of `scene_id` of the previous picture in output order, this indicates that the target pictures and the previous picture (in output order) are considered by the encoder to have been from different source scenes.

The value of scene_id shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of scene_id in the range of 0 to 255, inclusive, and in the range of 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of scene_id in the range of 256 to 511, inclusive, and in the range of 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of scene_id in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

NOTE 5 – When the first picture picA, in decoding order, of the CLVS vidSeqA represents a different scene than the last picture, in output order, of the previous CLVS of the same layer and a scene information SEI message is associated with PicA, the scene_id value of that scene information SEI message should have a random value within the value ranges constrained above. Subsequent scene_id and second_scene_id values may be selected for example by incrementing the initial randomly selected scene_id value. Consequently, when concatenating vidSeqA to a CLVS vidSeqB of the same layer, accidental use of the same scene_id values in vidSeqA and vidSeqB is unlikely.

scene_transition_type specifies in which type of a scene transition (if any) the target pictures are involved. The valid values of scene_transition_type are specified in Table D.3.

Table D.3 – scene_transition_type values

Value	Description
0	No transition
1	Fade to black
2	Fade from black
3	Unspecified transition from or to constant colour
4	Dissolve
5	Wipe
6	Unspecified mixture of two scenes

When scene_transition_type is greater than 3, the target pictures include contents both from the scene labelled by its scene_id and the next scene, in output order, which is labelled by second_scene_id (see below). The term "the current scene" is used to indicate the scene labelled by scene_id. The term "the next scene" is used to indicate the scene labelled by second_scene_id. It is not required for any following picture, in output order, to be labelled with scene_id equal to second_scene_id of the current SEI message.

Scene transition types are specified as follows:

- "No transition" specifies that the target pictures are not involved in a gradual scene transition.
NOTE 6 – When two consecutive pictures in output order have scene_transition_type equal to 0 and different values of scene_id, a scene cut occurred between the two pictures.
- "Fade to black" indicates that the target pictures are part of a sequence of pictures, in output order, involved in a fade to black scene transition, i.e., the luma samples of the scene gradually approach zero and the chroma samples of the scene gradually approach 128.
NOTE 7 – When two pictures are labelled to belong to the same scene transition and their scene_transition_type is "Fade to black", the later one, in output order, is darker than the previous one.
- "Fade from black" indicates that the target pictures are part of a sequence of pictures, in output order, involved in a fade from black scene transition, i.e., the luma samples of the scene gradually diverge from zero and the chroma samples of the scene may gradually diverge from 128.
NOTE 8 – When two pictures are labelled to belong to the same scene transition and their scene_transition_type is "Fade from black", the later one in output order is lighter than the previous one.
- "Dissolve" indicates that the sample values of each target picture (before encoding) were generated by calculating a sum of co-located weighted sample values of a picture from the current scene and a picture from the next scene. The weight of the current scene gradually decreases from full level to zero level, whereas the weight of the next scene gradually increases from zero level to full level. When two pictures are labelled to belong to the same scene transition and their scene_transition_type is "Dissolve", the weight of the current scene for the later one, in output order, is less than the weight of the current scene for the previous one, and the weight of the next scene for the later one, in output order, is greater than the weight of the next scene for the previous one.
- "Wipe" indicates that some of the sample values of each target picture (before encoding) were generated by copying co-located sample values of a picture in the current scene and the remaining sample values of each target picture

(before encoding) were generated by copying co-located sample values of a picture in the next scene. When two pictures are labelled to belong to the same scene transition and their `scene_transition_type` is "Wipe", the number of samples copied from the next scene to the later picture in output order is greater than the number of samples copied from the next scene to the previous picture.

second_scene_id identifies the next scene in the gradual scene transition in which the target pictures are involved. The value of `second_scene_id` shall not be equal to the value of `scene_id`. The value of `second_scene_id` shall not be equal to the value of `scene_id` in the previous picture in output order. When the next picture in output order is marked with a value of `scene_transition_type` less than 4, and the value of `second_scene_id` is the same as the value of `scene_id` of the next picture in output order, this indicates that the encoder considers one of the source scenes for the target pictures and the source scene for the next picture (in output order) to have been the same scene. When the value of `second_scene_id` is not equal to the value of `scene_id` or `second_scene_id` (when present) of the next picture in output order, this indicates that the encoder considers the target pictures and the next picture (in output order) to have been from different source scenes.

When the value of `scene_id` of a picture is equal to the value of `scene_id` of the following picture in output order and the value of `scene_transition_type` in both of these pictures is less than 4, this indicates that the encoder considers the two pictures to have been from the same source scene. When the values of `scene_id`, `scene_transition_type` and `second_scene_id` (when present) of a picture are equal to the values of `scene_id`, `scene_transition_type` and `second_scene_id` (respectively) of the following picture in output order and the value of `scene_transition_type` is greater than 0, this indicates that the encoder considers the two pictures to have been from the same source gradual scene transition.

The value of `second_scene_id` shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of `second_scene_id` in the range of 0 to 255, inclusive, and in the range of 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of `second_scene_id` in the range of 256 to 511, inclusive, and in the range of 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of `second_scene_id` in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

D.3.10 Picture snapshot SEI message semantics

The picture snapshot SEI message indicates that the current picture is labelled for use as determined by the application as a still-image snapshot of the video content.

snapshot_id specifies a snapshot identification number. `snapshot_id` shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of `snapshot_id` in the range of 0 to 255, inclusive, and in the range of 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of `snapshot_id` in the range of 256 to 511, inclusive, and in the range of 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of `snapshot_id` in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

D.3.11 Progressive refinement segment start SEI message semantics

The progressive refinement segment start SEI message specifies the beginning of a set of consecutive coded pictures in the current layer in decoding order that consists of the current picture and a sequence of one or more subsequent pictures in the current layer that refine the quality of the current picture, rather than a representation of a continually moving scene.

Let `picA` be the current picture. The tagged set of consecutive coded pictures `refinementPicSet` in the current layer consists of, in decoding order, the next picture in the current layer after the current picture in decoding order, when present, followed by zero or more pictures in the current layer, including all subsequent pictures in the current layer up to but not including any subsequent picture `picB` in the current layer for which one of the following conditions is true:

- The picture `picB` starts a new CLVS of the current layer.
- The value of `pic_order_cnt_delta` is greater than 0 and the `PicOrderCntVal` of the picture `picB`, i.e., `PicOrderCnt(picB)` is greater than `PicOrderCnt(picA) + pic_order_cnt_delta`, where `PicOrderCnt(picB)` and `PicOrderCnt(picA)` are the `PicOrderCntVal` values of `picB` and `picA`, respectively, immediately after the invocation of the decoding process for picture order count for `picB`.
- The picture `picB` is associated with a progressive refinement segment end SEI message that has the same `progressive_refinement_id` as the one in this SEI message and also applies to the current layer is decoded.

The decoding order of pictures within `refinementPicSet` should be the same as their output order.

progressive_refinement_id specifies an identification number for the progressive refinement operation. `progressive_refinement_id` shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of `progressive_refinement_id` in the range of 0 to 255, inclusive, and in the range of 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of `progressive_refinement_id` in the range of 256 to 511, inclusive, and in the range of 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of `progressive_refinement_id` in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

pic_order_cnt_delta specifies the last picture in refinementPicSet in decoding order as follows:

- If **pic_order_cnt_delta** is equal to 0, the last picture in refinementPicSet in decoding order is the following picture:
 - If the CLVS contains one or more pictures in the current layer that follow the current picture in decoding order and are associated with a progressive refinement segment end SEI message that has the same **progressive_refinement_id** and also applies to the current layer, the last picture in refinementPicSet in decoding order is the first of these pictures in decoding order.
 - Otherwise, the last picture in refinementPicSet in decoding order is the last picture in the current layer within the CLVS in decoding order.
- Otherwise, the last picture in refinementPicSet in decoding order is the following picture:
 - If the CLVS contains one or more pictures in the current layer that follow the current picture in decoding order, that are associated with a progressive refinement segment end SEI message with the same **progressive_refinement_id** and applicable to the current layer, and that precede any picture **picC** in the current layer in the CLVS for which **PicOrderCnt(picC)** is greater than **PicOrderCnt(picA) + pic_order_cnt_delta**, where **PicOrderCnt(picC)** and **PicOrderCnt(picA)** are the **PicOrderCntVal** of the **picC** and **picA**, respectively, immediately after the invocation of the decoding process for picture order count for **picC**, the last picture in refinementPicSet in decoding order is the first of these pictures in decoding order.
 - Otherwise, if the CLVS contains one or more pictures **picD** in the current layer that follow the current picture in decoding order for which **PicOrderCnt(picD)** is greater than **PicOrderCnt(picA) + pic_order_cnt_delta**, where **PicOrderCnt(picD)** and **PicOrderCnt(picA)** are the **PicOrderCntVal** values of **picD** and **picA**, respectively, immediately after the invocation of the decoding process for picture order count for **picD**, the last picture in refinementPicSet in decoding order is the last picture in the current layer that precedes the first of these pictures in decoding order.
 - Otherwise, the last picture in refinementPicSet in decoding order is the last picture in the current layer within the CLVS in decoding order.

The value of **pic_order_cnt_delta** shall be in the range of 0 to 256, inclusive.

D.3.12 Progressive refinement segment end SEI message semantics

The progressive refinement segment end SEI message specifies the end of a set of consecutive coded pictures that has been labelled by use of a progressive refinement segment start SEI message as an initial picture followed by a sequence of one or more pictures of the refinement of the quality of the initial picture and ending with the current picture.

progressive_refinement_id specifies an identification number for the progressive refinement operation. **progressive_refinement_id** shall be in the range of 0 to $2^{32} - 2$, inclusive.

The progressive refinement segment end SEI message specifies the end of any progressive refinement segment previously started using a progressive refinement segment start SEI message with the same value of **progressive_refinement_id**.

Values of **progressive_refinement_id** in the range of 0 to 255, inclusive, and in the range of 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of **progressive_refinement_id** in the range of 256 to 511, inclusive, and in the range of 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of **progressive_refinement_id** in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

D.3.13 Film grain characteristics SEI message semantics

This SEI message provides the decoder with a parameterized model for film grain synthesis.

NOTE 1 – For example, an encoder could use the film grain characteristics SEI message to characterize film grain that was present in the original source video material and was removed by pre-processing filtering techniques. Synthesis of simulated film grain on the decoded images for the display process is optional and does not need to exactly follow the specified semantics of the film grain characteristics SEI message. When synthesis of simulated film grain on the decoded images for the display process is performed, there is no requirement that the method by which the synthesis is performed be the same as the parameterized model for the film grain as provided in the film grain characteristics SEI message.

NOTE 2 – The display process is not specified in this Specification.

NOTE 3 – SMPTE RDD 5 specifies a film grain simulator based on the information provided in the film grain characteristics SEI message.

The film grain models specified in the film grain characteristics SEI message are expressed for application to decoded pictures that have 4:4:4 colour format with luma and chroma bit depths corresponding to the luma and chroma bit depths of the film grain model and use the same colour representation domain as the identified film grain model. When the colour format of the decoded video is not 4:4:4 or the decoded video uses a different luma or chroma bit depth from that of the film grain model or uses a different colour representation domain from that of the identified film grain model, an unspecified conversion process is expected to be applied to convert the decoded pictures to the form that is expressed for application of the film grain model.

NOTE 4 – Because the use of a specific method is not required for performing the film grain generation function used by the display process, a decoder could, if desired, down-convert the model information for chroma in order to simulate film grain for other chroma formats (4:2:0 or 4:2:2) rather than up-converting the decoded video (using a method not specified in this Specification) before performing film grain generation.

film_grain_characteristics_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous film grain characteristics SEI message in output order that applies to the current layer. **film_grain_characteristics_cancel_flag** equal to 0 indicates that film grain modelling information follows.

film_grain_model_id identifies the film grain simulation model as specified in Table D.4. The value of **film_grain_model_id** shall be in the range of 0 to 1, inclusive. The values of 2 and 3 for **film_grain_model_id** are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore film grain characteristic SEI messages with **film_grain_model_id** equal to 2 or 3.

Table D.4 – film_grain_model_id values

Value	Description
0	Frequency filtering
1	Auto-regression

separate_colour_description_present_flag equal to 1 indicates that a distinct combination of luma bit depth, chroma bit depth, video full range flag, colour primaries, transfer characteristics, and matrix coefficients for the film grain characteristics specified in the SEI message is present in the film grain characteristics SEI message syntax. **separate_colour_description_present_flag** equal to 0 indicates that the combination of luma bit depth, chroma bit depth, video full range flag, colour primaries, transfer characteristics, and matrix coefficients for the film grain characteristics specified in the SEI message are the same as indicated in the VUI parameters for the CVS.

NOTE 5 – When **separate_colour_description_present_flag** is equal to 1, any of the luma bit depth, chroma bit depth, video full range flag, colour primaries, transfer characteristics, and matrix coefficients specified for the film grain characteristics specified in the SEI message could differ from those for the pictures in the CVS.

When VUI parameters are not present for the CVS or the value of **colour_description_present_flag** is equal to 0, and equivalent information to that conveyed when **colour_description_present_flag** is equal to 1 is not conveyed by external means, **separate_colour_description_present_flag** shall be equal to 1.

The decoded image I_{decoded} used in the equations in this clause is in the same colour representation domain as the simulated film grain signal. Therefore, when any of these parameters does differ from that for the pictures in the CVS, the decoded image I_{decoded} used in the equations in this clause would be in a different colour representation domain than that for the pictures in the CVS. For example, when the value of **film_grain_bit_depth_luma_minus8** + 8 is greater than the bit depth of the luma component of the pictures in the CVS, the bit depth of I_{decoded} used in the equations in this clause is also greater than the bit depth of the luma component of the pictures in the CVS. In such a case, the decoded image I_{decoded} corresponding to an actual decoded picture would be generated by converting the actual decoded picture to be in the same colour representation domain as the simulated film grain signal. The process for converting the actual decoded pictures to the 4:4:4 colour format with same colour representation domain as the simulated film grain signal is not specified in this Specification.

film_grain_bit_depth_luma_minus8 plus 8 specifies the bit depth used for the luma component of the film grain characteristics specified in the SEI message. When **film_grain_bit_depth_luma_minus8** is not present in the film grain characteristics SEI message, the value of **film_grain_bit_depth_luma_minus8** is inferred to be equal to **bit_depth_luma_minus8**.

The value of **filmGrainBitDepth[0]** is derived as follows:

$$\text{filmGrainBitDepth}[0] = \text{film_grain_bit_depth_luma_minus8} + 8 \quad (\text{D-3})$$

film_grain_bit_depth_chroma_minus8 plus 8 specifies the bit depth used for the Cb and Cr components of the film grain characteristics specified in the SEI message. When **film_grain_bit_depth_chroma_minus8** is not present in the film grain characteristics SEI message, the value of **film_grain_bit_depth_chroma_minus8** is inferred to be equal to **bit_depth_chroma_minus8**.

The value of **filmGrainBitDepth[c]** for $c = 1$ and 2 is derived as follows:

$$\text{filmGrainBitDepth}[c] = \text{film_grain_bit_depth_chroma_minus8} + 8, \text{ with } c = 1, 2 \quad (\text{D-4})$$

film_grain_full_range_flag has the same semantics as specified in clause E.3.1 for the **video_full_range_flag** syntax element, except as follows:

- **film_grain_full_range_flag** specifies the video full range flag of the film grain characteristics specified in the SEI message, rather than the video full range flag used for the CVS.
- When **film_grain_full_range_flag** is not present in the film grain characteristics SEI message, the value of **film_grain_full_range_flag** is inferred to be equal to **video_full_range_flag**.

film_grain_colour_primaries has the same semantics as specified in clause E.3.1 for the **colour_primaries** syntax element, except as follows:

- **film_grain_colour_primaries** specifies the colour primaries of the film grain characteristics specified in the SEI message, rather than the colour primaries used for the CVS.
- When **film_grain_colour_primaries** is not present in the film grain characteristics SEI message, the value of **film_grain_colour_primaries** is inferred to be equal to **colour_primaries**.

film_grain_transfer_characteristics has the same semantics as specified in clause E.3.1 for the **transfer_characteristics** syntax element, except as follows:

- **film_grain_transfer_characteristics** specifies the transfer characteristics of the film grain characteristics specified in the SEI message, rather than the transfer characteristics used for the CVS.
- When **film_grain_transfer_characteristics** is not present in the film grain characteristics SEI message, the value of **film_grain_transfer_characteristics** is inferred to be equal to **transfer_characteristics**.

film_grain_matrix_coeffs has the same semantics as specified in clause E.3.1 for the **matrix_coeffs** syntax element, except as follows:

- **film_grain_matrix_coeffs** specifies the matrix coefficients of the film grain characteristics specified in the SEI message, rather than the matrix coefficients used for the CVS.
- When **film_grain_matrix_coeffs** is not present in the film grain characteristics SEI message, the value of **film_grain_matrix_coeffs** is inferred to be equal to **matrix_coeffs**.
- The values allowed for **film_grain_matrix_coeffs** are not constrained by the chroma format of the decoded pictures that is indicated by the value of **chroma_format_idc** for the semantics of the VUI parameters.

blending_mode_id identifies the blending mode used to blend the simulated film grain with the decoded images as specified in Table D.5. **blending_mode_id** shall be in the range of 0 to 1, inclusive. The values of 2 and 3 for **blending_mode_id** are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore film grain characteristic SEI messages with **blending_mode_id** equal to 2 or 3.

Table D.5 – blending_mode_id values

Value	Description
0	Additive
1	Multiplicative

Depending on the value of **blending_mode_id**, the blending mode is specified as follows:

- If **blending_mode_id** is equal to 0, the blending mode is additive as specified by:

$$I_{\text{grain}}[c][x][y] = \text{Clip3}(0, (1 \ll \text{filmGrainBitDepth}[c]) - 1, I_{\text{decoded}}[c][x][y] + G[c][x][y]) \quad (\text{D-5})$$

- Otherwise (**blending_mode_id** is equal to 1), the blending mode is multiplicative as specified by:

$$I_{\text{grain}}[c][x][y] = \text{Clip3}(0, (1 \ll \text{filmGrainBitDepth}[c]) - 1, I_{\text{decoded}}[c][x][y] + \text{Round}((I_{\text{decoded}}[c][x][y] * G[c][x][y]) \div ((1 \ll \text{bitDepth}[c]) - 1))) \quad (\text{D-6})$$

where $I_{\text{decoded}}[c][x][y]$ represents the sample value at coordinates x, y of the colour component c of the decoded image I_{decoded} , $G[c][x][y]$ is the simulated film grain value at the same position and colour component, and $\text{filmGrainBitDepth}[c]$ is the number of bits used for each sample in a fixed-length unsigned binary representation of the arrays $I_{\text{grain}}[c][x][y]$, $I_{\text{decoded}}[c][x][y]$, and $G[c][x][y]$, where $c = 0..2$, $x = 0..\text{pic_width_in_luma_samples} - 1$, and $y = 0..\text{pic_height_in_luma_samples} - 1$.

log2_scale_factor specifies a scale factor used in the film grain characterization equations.

comp_model_present_flag[c] equal to 0 indicates that film grain is not modelled on the c-th colour component, where c equal to 0 refers to the luma component, c equal to 1 refers to the Cb component, and c equal to 2 refers to the Cr component. **comp_model_present_flag[c]** equal to 1 indicates that syntax elements specifying modelling of film grain on colour component c are present in the SEI message.

When **separate_colour_description_present_flag** is equal to 0 and **chroma_format_idc** is equal to 0, the value of **comp_model_present_flag[1]** and **comp_model_present_flag[2]** shall be equal to 0.

num_intensity_intervals_minus1[c] plus 1 specifies the number of intensity intervals for which a specific set of model values has been estimated.

NOTE 6 – The intensity intervals could overlap in order to simulate multi-generational film grain.

num_model_values_minus1[c] plus 1 specifies the number of model values present for each intensity interval in which the film grain has been modelled. The value of **num_model_values_minus1[c]** shall be in the range of 0 to 5, inclusive.

intensity_interval_lower_bound[c][i] specifies the lower bound of the i-th intensity interval for which the set of model values applies.

intensity_interval_upper_bound[c][i] specifies the upper bound of the i-th intensity interval for which the set of model values applies.

The variable **intensityIntervalIdx[c][x][y][j]** represents the j-th index to the list of intensity intervals selected for the sample value $I_{\text{decoded}}[c][x][y]$ for $c = 0..2$, $x = 0..pic_width_in_luma_samples - 1$, $y = 0..pic_height_in_luma_samples - 1$, and $j = 0..numApplicableIntensityIntervals[c][x][y] - 1$, where **numApplicableIntensityIntervals[c][x][y]** is derived below.

Depending on the value of **film_grain_model_id**, the selection of one or more intensity intervals for the sample value $I_{\text{decoded}}[c][x][y]$ is specified as follows:

- The variable **numApplicableIntensityIntervals[c][x][y]** is initially set equal to 0.
- If **film_grain_model_id** is equal to 0, the following applies:
 - The top-left sample location (xB, yB) of the current 8x8 block b that contains the sample value $I_{\text{decoded}}[c][x][y]$ is derived as $(xB, yB) = (x / 8, y / 8)$.
 - The average value b_{avg} of the current 8x8 block b is derived as follows:

$$\begin{aligned}
 & \text{sum8x8} = 0 \\
 & \text{for}(i = 0; i < 8; i++) \\
 & \quad \text{for}(j = 0; j < 8; j++) \\
 & \quad \quad \text{sum8x8} += I_{\text{decoded}}[c][xB * 8 + i][yB * 8 + j] \\
 & b_{\text{avg}} = \text{Clip3}(0, 255, \\
 & \quad (\text{sum8x8} + (1 \ll (\text{filmGrainBitDepth}[c] - 3))) \gg (\text{filmGrainBitDepth}[c] - 2))
 \end{aligned} \tag{D-7}$$

- The values of **intensityIntervalIdx[c][x][y][j]** and **numApplicableIntensityIntervals[c][x][y]** are derived as follows:

$$\begin{aligned}
 & \text{for}(i = 0, j = 0; i \leq \text{num_intensity_intervals_minus1}[c]; i++) \\
 & \quad \text{if}(b_{\text{avg}} \geq \text{intensity_interval_lower_bound}[c][i] \\
 & \quad \quad \&\& b_{\text{avg}} \leq \text{intensity_interval_upper_bound}[c][i]) \{ \\
 & \quad \quad \text{intensityIntervalIdx}[c][x][y][j] = i \\
 & \quad \quad j++ \\
 & \quad \} \\
 & \text{numApplicableIntensityIntervals}[c][x][y] = j
 \end{aligned} \tag{D-8}$$

- Otherwise (**film_grain_model_id** is equal to 1), the values of **intensityIntervalIdx[c][x][y][j]** and **numApplicableIntensityIntervals[c][x][y]** are derived as follows:

$$\begin{aligned}
 & I_8[c][x][y] = (\text{filmGrainBitDepth}[c] == 8) ? (I_{\text{decoded}}[c][x][y] : \\
 & \quad \text{Clip3}(0, 255, (I_{\text{decoded}}[c][x][y] + \\
 & \quad \quad (1 \ll (\text{filmGrainBitDepth}[c] - 9))) \gg (\text{filmGrainBitDepth}[c] - 8))) \\
 & \text{for}(i = 0, j = 0; i \leq \text{num_intensity_intervals_minus1}[c]; i++) \\
 & \quad \text{if}(I_8[c][x][y] \geq \text{intensity_interval_lower_bound}[c][i] \&\& \\
 & \quad \quad I_8[c][x][y] \leq \text{intensity_interval_upper_bound}[c][i]) \{ \\
 & \quad \quad \text{intensityIntervalIdx}[c][x][y][j] = i \\
 & \quad \quad j++ \\
 & \quad \}
 \end{aligned} \tag{D-9}$$

}
 numApplicableIntensityIntervals[c][x][y] = j

Samples that do not fall into any of the defined intervals (i.e., those samples for which the value of numApplicableIntensityIntervals[c][x][y] is equal to 0) are not modified by the grain generation function. Samples that fall into more than one interval (i.e., those samples for which the value of numApplicableIntensityIntervals[c][x][y] is greater than 1) will originate multi-generation grain. Multi-generation grain results from adding the grain computed independently for each of the applicable intensity intervals.

In the equations in the remainder of this clause, the variable s_j in each instance of the list comp_model_value[c][s_j] is the value of intensityIntervalIdx[c][x][y][j] derived for the sample value $I_{\text{decoded}}[c][x][y]$.

comp_model_value[c][i][j] specifies the j -th model value present for the colour component c and the i -th intensity interval. The set of model values has different meaning depending on the value of film_grain_model_id.

The value of comp_model_value[c][i][j] is constrained as follows, and could be additionally constrained as specified elsewhere in this clause:

- If film_grain_model_id is equal to 0, comp_model_value[c][i][j] shall be in the range of 0 to $2^{\text{filmGrainBitDepth}[c]} - 1$, inclusive.
- Otherwise (film_grain_model_id is equal to 1), comp_model_value[c][i][j] shall be in the range of $-2^{(\text{filmGrainBitDepth}[c] - 1)}$ to $2^{(\text{filmGrainBitDepth}[c] - 1)} - 1$, inclusive.

Depending on the value of film_grain_model_id, the synthesis of the film grain is modelled as follows:

- If film_grain_model_id is equal to 0, a frequency filtering model enables simulating the original film grain for $c = 0..2$, $x = 0..\text{pic_width_in_luma_samples} - 1$ and $y = 0..\text{pic_height_in_luma_samples} - 1$ as specified by:

$$G[c][x][y] = (\text{comp_model_value}[c][s_j][0] * Q[c][x][y] + \text{comp_model_value}[c][s_j][5] * G[c - 1][x][y]) \gg \log2_scale_factor \quad (\text{D-10})$$

where $Q[c]$ is a two-dimensional random process generated by filtering 16×16 blocks gaussRv with random variable elements gaussRv_{ij} generated with a normalized Gaussian distribution (independent and identically distributed Gaussian random variable samples with zero mean and unity variance) and the value of an element $G[c - 1][x][y]$ used in the right-hand side of the equation is inferred to be equal to 0 when $c - 1$ is less than 0.

NOTE 7 – A normalized Gaussian random variable can be generated from two independent, uniformly distributed random variables over the interval from 0 to 1 (and not equal to 0), denoted as uRv₀ and uRv₁, using the Box-Muller transformation specified by:

$$\text{gaussRv}_{ij} = \text{Sqrt}(-2 * \text{Ln}(\text{uRv}_0)) * \text{Cos}(2 * \pi * \text{uRv}_1) \quad (\text{D-11})$$

where π is Archimedes' constant 3.141 592 653 589 793....

The band-pass filtering of blocks gaussRv can be performed in the discrete cosine transform (DCT) domain as follows:

```
for( y = 0; y < 16; y++ )
  for( x = 0; x < 16; x++ )
    if( ( x < comp_model_value[ c ][ s_j ][ 3 ] && y < comp_model_value[ c ][ s_j ][ 4 ] ) ||
        x > comp_model_value[ c ][ s_j ][ 1 ] || y > comp_model_value[ c ][ s_j ][ 2 ] )
      gaussRv[ x ][ y ] = 0
  filteredRv = IDCT16x16( gaussRv )
```

where IDCT16x16(z) refers to a unitary inverse discrete cosine transformation (IDCT) operating on a 16×16 matrix argument z as specified by:

$$\text{IDCT16x16}(z) = r * z * r^T \quad (\text{D-13})$$

where the superscript T indicates a matrix transposition and r is the 16×16 matrix with elements r_{ij} specified by:

$$r_{i,j} = \frac{((i == 0) ? 1 : \text{Sqrt}(2))}{4} * \text{Cos}\left(\frac{i * (2 * j + 1) * \pi}{32}\right) \quad (\text{D-14})$$

where π is Archimedes' constant 3.141 592 653 589 793....

$Q[c]$ is formed by the frequency-filtered blocks filteredRv.

NOTE 8 – Coded model values are based on blocks of size 16x16, but a decoder implementation could use other block sizes. For example, decoders implementing the IDCT on 8x8 blocks could down-convert by a factor of two the set of coded model values $\text{comp_model_value}[c][s_j][i]$ for i equal to 1..4.

NOTE 9 – To reduce the degree of visible blocks that can result from mosaicking the frequency-filtered blocks filteredRv, decoders could apply a low-pass filter to the boundaries between frequency-filtered blocks.

- Otherwise ($\text{film_grain_model_id}$ is equal to 1), an auto-regression model enables simulating the original film grain for $c = 0..2$, $x = 0..\text{pic_width_in_luma_samples} - 1$, and $y = 0..\text{pic_height_in_luma_samples} - 1$ as specified by:

$$\begin{aligned} G[c][x][y] = & (\text{comp_model_value}[c][s_j][0] * n[c][x][y] + \\ & \text{comp_model_value}[c][s_j][1] * (G[c][x-1][y] + ((\text{comp_model_value}[c][s_j][4] * \\ & G[c][x][y-1]) >> \log_2_scale_factor)) + \\ & \text{comp_model_value}[c][s_j][3] * (((\text{comp_model_value}[c][s_j][4] * G[c][x-1][y-1]) >> \\ & \log_2_scale_factor) + G[c][x+1][y-1]) + \\ & \text{comp_model_value}[c][s_j][5] * (G[c][x-2][y] + \\ & ((\text{comp_model_value}[c][s_j][4] * \text{comp_model_value}[c][s_j][4] * G[c][x][y-2]) >> \\ & (2 * \log_2_scale_factor))) + \\ & \text{comp_model_value}[c][s_j][2] * G[c-1][x][y]) >> \log_2_scale_factor \end{aligned} \quad (D-15)$$

where $n[c][x][y]$ is a random variable with normalized Gaussian distribution (independent and identically distributed Gaussian random variable samples with zero mean and unity variance for each value of c , x , and y) and the value of an element $G[c][x][y]$ used in the right-hand side of the equation is inferred to be equal to 0 when any of the following conditions are true:

- x is less than 0,
- y is less than 0,
- c is less than 0.

$\text{comp_model_value}[c][i][0]$ provides the first model value for the model as specified by $\text{film_grain_model_id}$. $\text{comp_model_value}[c][i][0]$ corresponds to the standard deviation of the Gaussian noise term in the generation functions specified in Equations D-10 through D-15.

$\text{comp_model_value}[c][i][1]$ provides the second model value for the model as specified by $\text{film_grain_model_id}$. When $\text{film_grain_model_id}$ is equal to 0, $\text{comp_model_value}[c][i][1]$ shall be greater than or equal to 0 and less than 16.

When not present in the film grain characteristics SEI message, $\text{comp_model_value}[c][i][1]$ is inferred as follows:

- If $\text{film_grain_model_id}$ is equal to 0, $\text{comp_model_value}[c][i][1]$ is inferred to be equal to 8.
- Otherwise ($\text{film_grain_model_id}$ is equal to 1), $\text{comp_model_value}[c][i][1]$ is inferred to be equal to 0.

$\text{comp_model_value}[c][i][1]$ is interpreted as follows:

- If $\text{film_grain_model_id}$ is equal to 0, $\text{comp_model_value}[c][i][1]$ indicates the horizontal high cut frequency to be used to filter the DCT of a block of 16x16 random values.
- Otherwise ($\text{film_grain_model_id}$ is equal to 1), $\text{comp_model_value}[c][i][1]$ indicates the first order spatial correlation for neighbouring samples at positions $(x-1, y)$ and $(x, y-1)$.

$\text{comp_model_value}[c][i][2]$ provides the third model value for the model as specified by $\text{film_grain_model_id}$. When $\text{film_grain_model_id}$ is equal to 0, $\text{comp_model_value}[c][i][2]$ shall be greater than or equal to 0 and less than 16.

When not present in the film grain characteristics SEI message, $\text{comp_model_value}[c][i][2]$ is inferred as follows:

- If $\text{film_grain_model_id}$ is equal to 0, $\text{comp_model_value}[c][i][2]$ is inferred to be equal to $\text{comp_model_value}[c][i][1]$
- Otherwise ($\text{film_grain_model_id}$ is equal to 1), $\text{comp_model_value}[c][i][2]$ is inferred to be equal to 0.

$\text{comp_model_value}[c][i][2]$ is interpreted as follows:

- If $\text{film_grain_model_id}$ is equal to 0, $\text{comp_model_value}[c][i][2]$ indicates the vertical high cut frequency to be used to filter the DCT of a block of 16x16 random values.
- Otherwise ($\text{film_grain_model_id}$ is equal to 1), $\text{comp_model_value}[c][i][2]$ indicates the colour correlation between consecutive colour components.

`comp_model_value[c][i][3]` provides the fourth model value for the model as specified by `film_grain_model_id`. When `film_grain_model_id` is equal to 0, `comp_model_value[c][i][3]` shall be greater than or equal to 0 and less than or equal to `comp_model_value[c][i][1]`.

When not present in the film grain characteristics SEI message, `comp_model_value[c][i][3]` is inferred to be equal to 0.

`comp_model_value[c][i][3]` is interpreted as follows:

- If `film_grain_model_id` is equal to 0, `comp_model_value[c][i][3]` indicates the horizontal low cut frequency to be used to filter the DCT of a block of 16x16 random values.
- Otherwise (`film_grain_model_id` is equal to 1), `comp_model_value[c][i][3]` indicates the first order spatial correlation for neighbouring samples at positions $(x - 1, y - 1)$ and $(x + 1, y - 1)$.

`comp_model_value[c][i][4]` provides the fifth model value for the model as specified by `film_grain_model_id`. When `film_grain_model_id` is equal to 0, `comp_model_value[c][i][4]` shall be greater than or equal to 0 and less than or equal to `comp_model_value[c][i][2]`.

When not present in the film grain characteristics SEI message, `comp_model_value[c][i][4]` is inferred to be equal to `film_grain_model_id`.

`comp_model_value[c][i][4]` is interpreted as follows:

- If `film_grain_model_id` is equal to 0, `comp_model_value[c][i][4]` indicates the vertical low cut frequency to be used to filter the DCT of a block of 16x16 random values.
- Otherwise (`film_grain_model_id` is equal to 1), `comp_model_value[c][i][4]` indicates the aspect ratio of the modelled grain.

`comp_model_value[c][i][5]` provides the sixth model value for the model as specified by `film_grain_model_id`.

When not present in the film grain characteristics SEI message, `comp_model_value[c][i][5]` is inferred to be equal to 0.

`comp_model_value[c][i][5]` is interpreted as follows:

- If `film_grain_model_id` is equal to 0, `comp_model_value[c][i][5]` indicates the colour correlation between consecutive colour components.
- Otherwise (`film_grain_model_id` is equal to 1), `comp_model_value[c][i][5]` indicates the second order spatial correlation for neighbouring samples at positions $(x, y - 2)$ and $(x - 2, y)$.

film_grain_characteristics_persistence_flag specifies the persistence of the film grain characteristics SEI message for the current layer.

`film_grain_characteristics_persistence_flag` equal to 0 specifies that the film grain characteristics SEI message applies to the current decoded picture only.

Let `picA` be the current picture. `film_grain_characteristics_persistence_flag` equal to 1 specifies that the film grain characteristics SEI message persists for the current layer in output order until any of the following conditions are true:

- A new CLVS of the current layer begins.
- The bitstream ends.
- A picture `picB` in the current layer in an access unit containing a film grain characteristics SEI message that is applicable to the current layer is output for which `PicOrderCnt(picB)` is greater than `PicOrderCnt(picA)`, where `PicOrderCnt(picB)` and `PicOrderCnt(picA)` are the `PicOrderCntVal` values of `picB` and `picA`, respectively, immediately after the invocation of the decoding process for picture order count for `picB`.

D.3.14 Post-filter hint SEI message semantics

This SEI message provides the coefficients of a post-filter or correlation information for the design of a post-filter for potential use in post-processing of the current picture after it is decoded and output to obtain improved displayed quality.

filter_hint_size_y specifies the vertical size of the filter coefficient or correlation array. The value of `filter_hint_size_y` shall be in the range of 1 to 15, inclusive.

filter_hint_size_x specifies the horizontal size of the filter coefficient or correlation array. The value of `filter_hint_size_x` shall be in the range of 1 to 15, inclusive.

filter_hint_type identifies the type of the transmitted filter hints as specified in Table D.6. The value of `filter_hint_type` shall be in the range of 0 to 2, inclusive. The value of `filter_hint_type` equal to 3 is reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore post-filter hint SEI messages having `filter_hint_type` equal to 3.

Table D.6 – filter_hint_type values

Value	Description
0	Coefficients of a 2D-FIR filter
1	Coefficients of two 1D-FIR filters
2	Cross-correlation matrix

filter_hint_value[cIdx][cy][cx] specifies a filter coefficient or an element of a cross-correlation matrix between the original and the decoded signal with 16-bit precision. The value of **filter_hint_value**[cIdx][cy][cx] shall be in the range of $-2^{31} + 1$ to $2^{31} - 1$, inclusive. cIdx specifies the related colour component, cy represents a counter in vertical direction and cx represents a counter in horizontal direction. Depending on the value of **filter_hint_type**, the following applies:

- If **filter_hint_type** is equal to 0, the coefficients of a 2-dimensional finite impulse response (FIR) filter with the size of **filter_hint_size_y** * **filter_hint_size_x** are transmitted.
- Otherwise, if **filter_hint_type** is equal to 1, the filter coefficients of two 1-dimensional FIR filters are transmitted. In this case, **filter_hint_size_y** shall be equal to 2. The index cy equal to 0 specifies the filter coefficients of the horizontal filter and cy equal to 1 specifies the filter coefficients of the vertical filter. In the filtering process, the horizontal filter is applied first and the result is filtered by the vertical filter.
- Otherwise (**filter_hint_type** is equal to 2), the transmitted hints specify a cross-correlation matrix between the original signal *s* and the decoded signal *s'*.

NOTE 1 – The normalized cross-correlation matrix for a related colour component identified by cIdx with the size of **filter_hint_size_y** * **filter_hint_size_x** is defined as follows:

$$\text{filter_hint_value}(\text{cIdx}, \text{cy}, \text{cx}) = \frac{1}{(2^{8+\text{bitDepth}} - 1)^2 * h * w} \sum_{m=0}^{h-1} \sum_{n=0}^{w-1} s(m, n) * s'(m + \text{cy} - \text{OffsetY}, n + \text{cx} - \text{OffsetX}) \quad (\text{D-16})$$

where *s* denotes array of samples of the colour component cIdx of the original picture, *s'* denotes corresponding array of the decoded picture, *h* denotes the vertical height of the related colour component, *w* denotes the horizontal width of the related colour component, bitDepth denotes the bit depth of the colour component, OffsetY is equal to (**filter_hint_size_y** >> 1), OffsetX is equal to (**filter_hint_size_x** >> 1), $0 \leq \text{cy} < \text{filter_hint_size_y}$ and $0 \leq \text{cx} < \text{filter_hint_size_x}$.

NOTE 2 – A decoder can derive a Wiener post-filter from the cross-correlation matrix of original and decoded signal and the auto-correlation matrix of the decoded signal.

D.3.15 Tone mapping information SEI message semantics

This SEI message provides information to enable remapping of the colour samples of the output decoded pictures for customization to particular display environments. The remapping process maps coded sample values in the RGB colour space (specified in Annex E) to target sample values. The mappings are expressed either in the luma or RGB colour space domain and should be applied to the luma component or to each RGB component produced by colour space conversion of the decoded image accordingly.

tone_map_id contains an identifying number that may be used to identify the purpose of the tone mapping model. The value of **tone_map_id** shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of **tone_map_id** from 0 to 255, inclusive, and from 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of **tone_map_id** from 256 to 511, inclusive, and from 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of **tone_map_id** in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

NOTE 1 – The **tone_map_id** can be used to support tone mapping operations that are suitable for different display scenarios. For example, different values of **tone_map_id** may correspond to different display bit depths.

tone_map_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous tone mapping information SEI message in output order that applies to the current layer. **tone_map_cancel_flag** equal to 0 indicates that tone mapping information follows.

tone_map_persistence_flag specifies the persistence of the tone mapping information SEI message.

tone_map_persistence_flag equal to 0 specifies that the tone mapping information applies to the current decoded picture only.

Let `picA` be the current picture. `tone_map_persistence_flag` equal to 1 specifies that the tone mapping information persists for the current layer in output order until any of the following conditions are true:

- A new CLVS of the current layer begins.
- A picture `picB` in the current layer in an access unit containing a tone mapping information SEI message with the same value of `tone_map_id` and applicable to the current layer is output for which `PicOrderCnt(picB)` is greater than `PicOrderCnt(picA)`, where `PicOrderCnt(picB)` and `PicOrderCnt(picA)` are the `PicOrderCntVal` values of `picB` and `picA`, respectively, immediately after the invocation of the decoding process for picture order count for `picB`.

coded_data_bit_depth specifies the `BitDepthY` for interpretation of the luma component of the associated pictures for purposes of interpretation of the tone mapping information SEI message. When tone mapping information SEI messages are present that have `coded_data_bit_depth` that is not equal to `BitDepthY`, these refer to the hypothetical result of a transcoding operation performed to convert the coded video to the `BitDepthY` corresponding to the value of `coded_data_bit_depth`.

The value of `coded_data_bit_depth` shall be in the range of 8 to 14, inclusive. Values of `coded_data_bit_depth` from 0 to 7 and from 15 to 255 are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore all tone mapping SEI messages that contain a `coded_data_bit_depth` in the range of 0 to 7, inclusive, or in the range of 15 to 255, inclusive, and bitstreams shall not contain such values.

target_bit_depth specifies the bit depth of the output of the dynamic range mapping function (or tone mapping function) described by the tone mapping information SEI message. The tone mapping function specified with a particular `target_bit_depth` is suggested to be reasonable for all display bit depths that are less than or equal to the `target_bit_depth`.

The value of `target_bit_depth` shall be in the range of 1 to 16, inclusive. Values of `target_bit_depth` equal to 0 and in the range of 17 to 255, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore all tone mapping SEI messages that contain a value of `target_bit_depth` equal to 0 or in the range of 17 to 255, inclusive, and bitstreams shall not contain such values.

tone_map_model_id specifies the model utilized for mapping the coded data into the `target_bit_depth` range. Values greater than 4 are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore all tone mapping SEI messages that contain a value of `tone_map_model_id` greater than 4 and bitstreams shall not contain such values.

NOTE 2 – A `tone_map_model_id` of 0 corresponds to a linear mapping with clipping; a `tone_map_model_id` of 1 corresponds to a sigmoidal mapping; a `tone_map_model_id` of 2 corresponds to a user-defined table mapping, and a `tone_map_model_id` of 3 corresponds to a piece-wise linear mapping, `tone_map_model_id` of 4 corresponds to luminance dynamic range information.

min_value specifies the RGB sample value that maps to the minimum value in the bit depth indicated by `target_bit_depth`. It is used in combination with the `max_value` parameter. All sample values in the decoded picture that are less than or equal to `min_value`, after conversion to RGB as necessary, are mapped to this minimum value in the `target_bit_depth` representation.

max_value specifies the RGB sample value that maps to the maximum value in the bit depth indicated by `target_bit_depth`. It is used in combination with the `min_value` parameter. All sample values in the decoded picture that are greater than or equal to `max_value`, after conversion to RGB as necessary, are mapped to this maximum value in the `target_bit_depth` representation.

When present, `max_value` shall be greater than or equal to `min_value`.

sigmoid_midpoint specifies the RGB sample value of the coded data that are mapped to the centre point of the `target_bit_depth` representation. It is used in combination with the `sigmoid_width` parameter.

sigmoid_width specifies the distance between two coded data values that approximately correspond to the 5% and 95% values of the `target_bit_depth` representation, respectively. It is used in combination with the `sigmoid_midpoint` parameter and is interpreted according to the following function:

$$f(i) = \text{Round} \left(\frac{2^{\text{target_bit_depth} - 1}}{1 + \exp \left(\frac{-6 * (i - \text{sigmoid_midpoint})}{\text{sigmoid_width}} \right)} \right) \quad (\text{D-17})$$

where $f(i)$ denotes the function that maps an RGB sample value i from the coded data to a resulting RGB sample value in the `target_bit_depth` representation.

start_of_coded_interval[i] specifies the beginning point of an interval in the coded data such that all RGB sample values that are greater than or equal to `start_of_coded_interval[i]` and less than `start_of_coded_interval[i + 1]` are mapped to i

in the target bit depth representation. The value of `start_of_coded_interval[2target_bit_depth]` is equal to `2coded_data_bit_depth`. The number of bits used for the representation of the `start_of_coded_interval` is $((\text{coded_data_bit_depth} + 7) \gg 3) \ll 3$.

num_pivots specifies the number of pivot points in the piece-wise linear mapping function without counting the two default end points, (0, 0) and $(2^{\text{coded_data_bit_depth}} - 1, 2^{\text{target_bit_depth}} - 1)$.

coded_pivot_value[i] specifies the value in the `coded_data_bit_depth` corresponding to the *i*-th pivot point. The number of bits used for the representation of the `coded_pivot_value` is $((\text{coded_data_bit_depth} + 7) \gg 3) \ll 3$.

target_pivot_value[i] specifies the value in the reference `target_bit_depth` corresponding to the *i*-th pivot point. The number of bits used for the representation of the `target_pivot_value` is $((\text{target_bit_depth} + 7) \gg 3) \ll 3$.

camera_iso_speed_idc indicates the camera ISO speed for daylight illumination as specified in ISO 12232, interpreted as specified in Table D.7. When `camera_iso_speed_idc` indicates EXTENDED_ISO, the ISO speed is indicated by `camera_iso_speed_value`.

camera_iso_speed_value indicates the camera ISO speed for daylight illumination as specified in ISO 12232 when `camera_iso_speed_idc` indicates EXTENDED_ISO. The value of `camera_iso_speed_value` shall not be equal to 0.

exposure_idx_idc indicates the exposure index setting of the camera as specified in ISO 12232, interpreted as specified in Table D.7. When `exposure_idx_idc` indicates EXTENDED_ISO, the exposure index is indicated by `exposure_idx_value`.

The values of `camera_iso_speed_idc` and `exposure_idx_idc` in the range of 31 to 254, inclusive, are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders conforming to this version of this Specification shall ignore tone mapping SEI messages that contain these values.

exposure_idx_value indicates the exposure index setting of the camera as specified in ISO 12232 when `exposure_idx_idc` indicates EXTENDED_ISO. The value of `exposure_idx_value` shall not be equal to 0.

Table D.7 – Interpretation of camera_iso_speed_idc and exposure_idx_idc

camera_iso_speed_idc or exposure_idx_idc	Indicated value
0	Unspecified
1	10
2	12
3	16
4	20
5	25
6	32
7	40
8	50
9	64
10	80
11	100
12	125
13	160
14	200
15	250
16	320
17	400
18	500
19	640
20	800
21	1 000
22	1 250
23	1 600
24	2 000
25	2 500
26	3 200
27	4 000
28	5 000
29	6 400
30	8 000
31..254	Reserved
255	EXTENDED_ISO

exposure_compensation_value_sign_flag, when applicable as specified below, specifies the sign of the variable ExposureCompensationValue that indicates the exposure compensation value setting used for the process of image production.

exposure_compensation_value_numerator, when applicable as specified below, specifies the numerator of the variable ExposureCompensationValue that indicates the exposure compensation value setting used for the process of image production.

exposure_compensation_value_denom_idc, when not equal to 0, specifies the denominator of the variable ExposureCompensationValue that indicates the exposure compensation value setting used for the process of image production.

When `exposure_compensation_value_denom_idc` is present and not equal to 0, the variable `ExposureCompensationValue` is derived from `exposure_compensation_value_sign_flag`, `exposure_compensation_value_numerator` and `exposure_compensation_value_denom_idc`. `exposure_compensation_value_sign_flag` equal to 0 indicates that the `ExposureCompensationValue` is positive. `exposure_compensation_value_sign_flag` equal to 1 indicates that the `ExposureCompensationValue` is negative. When `ExposureCompensationValue` is positive, the image is indicated to have been further sensitized through the process of production, relative to the recommended exposure index of the camera as specified in ISO 12232. When `ExposureCompensationValue` is negative, the image is indicated to have been further desensitized through the process of production, relative to the recommended exposure index of the camera as specified in ISO 12232.

When `exposure_compensation_value_denom_idc` is present and not equal to 0, the variable `ExposureCompensationValue` is derived as follows:

$$\text{ExposureCompensationValue} = (1 - 2 * \text{exposure_compensation_value_sign_flag}) * \frac{\text{exposure_compensation_value_numerator}}{\text{exposure_compensation_value_denom_idc}}$$

(D-18)

The value of `ExposureCompensationValue` is interpreted in units of exposure steps such that an increase of 1 in `ExposureCompensationValue` corresponds to a doubling of exposure in units of lux-seconds. For example, the exposure compensation value equal to $+1 \div 2$ at the production stage may be indicated by setting `exposure_compensation_value_sign_flag` to 0, `exposure_compensation_value_numerator` to 1 and `exposure_compensation_value_denom_idc` to 2.

When `exposure_compensation_value_denom_idc` is present and equal to 0, the exposure compensation value is indicated as unknown or unspecified.

ref_screen_luminance_white indicates the reference screen brightness setting for the extended white level used for image production process in units of candelas per square metre.

extended_range_white_level indicates the luminance dynamic range for extended dynamic-range display of the associated pictures, after conversion to the linear light domain for display, expressed as an integer percentage relative to the nominal white level. The value of `extended_range_white_level` should be greater than or equal to 100.

nominal_black_level_code_value indicates the luma sample value of the associated decoded pictures to which the nominal black level is assigned. For example, when `coded_data_bit_depth` is equal to 8, `video_full_range_flag` is equal to 0, and `matrix_coeffs` is equal to 1, `nominal_black_level_code_value` should be equal to 16.

nominal_white_level_code_value indicates the luma sample value of the associated decoded pictures to which the nominal white level is assigned. For example, when `coded_data_bit_depth` is equal to 8, `video_full_range_flag` is equal to 0 and `matrix_coeffs` is equal to 1, `nominal_white_level_code_value` should be equal to 235. When present, the value of `nominal_white_level_code_value` shall be greater than `nominal_black_level_code_value`.

extended_white_level_code_value indicates the luma sample value of the associated decoded pictures to which the white level associated with an extended dynamic range is assigned. When present, the value of `extended_white_level_code_value` shall be greater than or equal to `nominal_white_level_code_value`.

D.3.16 Frame packing arrangement SEI message semantics

This SEI message informs the decoder that the output cropped decoded picture contains samples of multiple distinct spatially packed constituent frames that are packed into one frame, or that the output cropped decoded pictures in output order form a temporal interleaving of alternating first and second constituent frames, using an indicated frame packing arrangement scheme. This information can be used by the decoder to appropriately rearrange the samples and process the samples of the constituent frames appropriately for display or other purposes (which are outside the scope of this Specification).

This SEI message may be associated with pictures that are either frames (when `field_seq_flag` is equal to 0) or fields (when `field_seq_flag` is equal to 1). The frame packing arrangement of the samples is specified in terms of the sampling structure of a frame in order to define a frame packing arrangement structure that is invariant with respect to whether a picture is a single field of such a packed frame or is a complete packed frame.

When `general_non_packed_constraint_flag` is equal to 1 for a CVS, there shall be no frame packing arrangement SEI messages in the CVS.

frame_packing_arrangement_id contains an identifying number that may be used to identify the usage of the frame packing arrangement SEI message. The value of `frame_packing_arrangement_id` shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of `frame_packing_arrangement_id` from 0 to 255, inclusive, and from 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of `frame_packing_arrangement_id` from 256 to 511, inclusive, and from 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of `frame_packing_arrangement_id` in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

`frame_packing_arrangement_cancel_flag` equal to 1 indicates that the SEI message cancels the persistence of any previous frame packing arrangement SEI message in output order that applies to the current layer. `frame_packing_arrangement_cancel_flag` equal to 0 indicates that frame packing arrangement information follows.

`frame_packing_arrangement_type` identifies the indicated interpretation of the sample arrays of the output cropped decoded picture as specified in Table D.8.

When `frame_packing_arrangement_type` is equal to 3 or 4, each component plane of the output cropped decoded picture contains all samples (when `field_pic_flag` is equal to 0) or the samples corresponding to the top or bottom field (when `field_pic_flag` is equal to 1) of the samples of a frame packing arrangement structure.

Table D.8 – Definition of `frame_packing_arrangement_type`

Value	Interpretation
3	The frame packing arrangement structure contains a side-by-side packing arrangement of corresponding planes of two constituent frames as illustrated in Figure D.4, Figure D.5 and Figure D.8.
4	The frame packing arrangement structure contains a top-bottom packing arrangement of corresponding planes of two constituent frames as illustrated in Figure D.6 and Figure D.7.
5	The component planes of the output cropped decoded pictures in output order form a temporal interleaving of alternating first and second constituent frames as illustrated in Figure D.9.

NOTE 1 – Figure D.4 to Figure D.8 provide typical examples of rearrangement and upconversion processing for various packing arrangement schemes. Actual characteristics of the constituent frames are signalled in detail by the subsequent syntax elements of the frame packing arrangement SEI message. In Figure D.4 to Figure D.8, an upconversion processing is performed on each constituent frame to produce frames having the same resolution as that of the decoded frame. An example of the upsampling method to be applied to a quincunx sampled frame as shown in Figure D.8 is to fill in missing positions with an average of the available spatially neighbouring samples (the average of the values of the available samples above, below, to the left and to the right of each sample to be generated). The actual upconversion process to be performed, if any, is outside the scope of this Specification.

NOTE 2 – When the output time of the samples of constituent frame 0 differs from the output time of the samples of constituent frame 1 (i.e., when `field_views_flag` is equal to 1 or `frame_packing_arrangement_type` is equal to 5) and the display system in use presents two views simultaneously, the display time for constituent frame 0 should be delayed to coincide with the display time for constituent frame 1. (The display process is not specified in this Specification.)

NOTE 3 – When `field_views_flag` is equal to 1 or `frame_packing_arrangement_type` is equal to 5, the value 0 for `fixed_pic_rate_within_cvs_flag` is not expected to be prevalent in industry use of this SEI message.

NOTE 4 – `frame_packing_arrangement_type` equal to 5 describes a temporal interleaving process of different views.

All other values of `frame_packing_arrangement_type` are reserved for future use by ITU-T | ISO/IEC. It is a requirement of bitstream conformance that bitstreams conforming to this version of this Specification shall not contain such other values of `frame_packing_arrangement_type`. Decoders shall ignore frame packing arrangement SEI messages that contain reserved values of `frame_packing_arrangement_type`.

`quincunx_sampling_flag` equal to 1 indicates that each colour component plane of each constituent frame is quincunx sampled as illustrated in Figure D.8 and `quincunx_sampling_flag` equal to 0 indicates that the colour component planes of each constituent frame are not quincunx sampled.

When `frame_packing_arrangement_type` is equal to 5, it is a requirement of bitstream conformance that `quincunx_sampling_flag` shall be equal to 0.

NOTE 5 – For any chroma format (4:2:0, 4:2:2 or 4:4:4), the luma plane and each chroma plane is quincunx sampled as illustrated in Figure D.8 when `quincunx_sampling_flag` is equal to 1.

`content_interpretation_type` indicates the intended interpretation of the constituent frames as specified in Table D.9. Values of `content_interpretation_type` that do not appear in Table D.9 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore frame packing arrangement SEI messages that contain reserved values of `content_interpretation_type`.

For each specified frame packing arrangement scheme, there are two constituent frames that are referred to as frame 0 and frame 1.

Table D.9 – Definition of content_interpretation_type

Value	Interpretation
0	Unspecified relationship between the frame packed constituent frames
1	Indicates that the two constituent frames form the left and right views of a stereo view scene, with frame 0 being associated with the left view and frame 1 being associated with the right view
2	Indicates that the two constituent frames form the right and left views of a stereo view scene, with frame 0 being associated with the right view and frame 1 being associated with the left view

NOTE 6 – The value 2 for content_interpretation_type is not expected to be prevalent in industry use of this SEI message. However, the value was specified herein for purposes of completeness.

spatial_flipping_flag equal to 1, when frame_packing_arrangement_type is equal to 3 or 4, indicates that one of the two constituent frames is spatially flipped relative to its intended orientation for display or other such purposes.

When frame_packing_arrangement_type is equal to 3 or 4 and spatial_flipping_flag is equal to 1, the type of spatial flipping that is indicated is as follows:

- If frame_packing_arrangement_type is equal to 3, the indicated spatial flipping is horizontal flipping.
- Otherwise (frame_packing_arrangement_type is equal to 4), the indicated spatial flipping is vertical flipping.

When frame_packing_arrangement_type is not equal to 3 or 4, it is a requirement of bitstream conformance that spatial_flipping_flag shall be equal to 0. When frame_packing_arrangement_type is not equal to 3 or 4, the value 1 for spatial_flipping_flag is reserved for future use by ITU-T | ISO/IEC. When frame_packing_arrangement_type is not equal to 3 or 4, decoders shall ignore the value 1 for spatial_flipping_flag.

frame0_flipped_flag, when spatial_flipping_flag is equal to 1, indicates which one of the two constituent frames is flipped.

When spatial_flipping_flag is equal to 1, frame0_flipped_flag equal to 0 indicates that frame 0 is not spatially flipped and frame 1 is spatially flipped and frame0_flipped_flag equal to 1 indicates that frame 0 is spatially flipped and frame 1 is not spatially flipped.

When spatial_flipping_flag is equal to 0, it is a requirement of bitstream conformance that frame0_flipped_flag shall be equal to 0. When spatial_flipping_flag is equal to 0, the value 1 for spatial_flipping_flag is reserved for future use by ITU-T | ISO/IEC. When spatial_flipping_flag is equal to 0, decoders shall ignore the value of frame0_flipped_flag.

field_views_flag equal to 1 indicates that all pictures in the current CVS are coded as fields, all fields of a particular parity are considered a first constituent frame and all fields of the opposite parity are considered a second constituent frame. It is a requirement of bitstream conformance that the field_views_flag shall be equal to 0, the value 1 for field_views_flag is reserved for future use by ITU-T | ISO/IEC and decoders shall ignore the value of field_views_flag.

current_frame_is_frame0_flag equal to 1, when frame_packing_arrangement_type is equal to 5, indicates that the current decoded frame is constituent frame 0 and the next decoded frame in output order is constituent frame 1 and the display time of the constituent frame 0 should be delayed to coincide with the display time of constituent frame 1. current_frame_is_frame0_flag equal to 0, when frame_packing_arrangement_type is equal to 5, indicates that the current decoded frame is constituent frame 1 and the previous decoded frame in output order is constituent frame 0 and the display time of the constituent frame 1 should not be delayed for purposes of stereo-view pairing.

When frame_packing_arrangement_type is not equal to 5, the constituent frame associated with the upper-left sample of the decoded frame is considered to be constituent frame 0 and the other constituent frame is considered to be constituent frame 1. When frame_packing_arrangement_type is not equal to 5, it is a requirement of bitstream conformance that current_frame_is_frame0_flag shall be equal to 0. When frame_packing_arrangement_type is not equal to 5, the value 1 for current_frame_is_frame0_flag is reserved for future use by ITU-T | ISO/IEC. When frame_packing_arrangement_type is not equal to 5, decoders shall ignore the value of current_frame_is_frame0_flag.

frame0_self_contained_flag equal to 1 indicates that no inter prediction operations within the decoding process for the samples of constituent frame 0 of the CVS refer to samples of any constituent frame 1. frame0_self_contained_flag equal to 0 indicates that some inter prediction operations within the decoding process for the samples of constituent frame 0 of the CVS may or may not refer to samples of some constituent frame 1. Within a CVS, the value of frame0_self_contained_flag in all frame packing arrangement SEI messages shall be the same.

frame1_self_contained_flag equal to 1 indicates that no inter prediction operations within the decoding process for the samples of constituent frame 1 of the CVS refer to samples of any constituent frame 0. **frame1_self_contained_flag** equal to 0 indicates that some inter prediction operations within the decoding process for the samples of constituent frame 1 of the CVS may or may not refer to samples of some constituent frame 0. Within a CVS, the value of **frame1_self_contained_flag** in all frame packing arrangement SEI messages shall be the same.

When **quincunx_sampling_flag** is equal to 0 and **frame_packing_arrangement_type** is not equal to 5, two (x, y) coordinate pairs are specified to determine the indicated luma sampling grid alignment for constituent frame 0 and constituent frame 1, relative to the upper left corner of the rectangular area represented by the samples of the corresponding constituent frame.

NOTE 7 – The location of chroma samples relative to luma samples can be indicated by the **chroma_sample_loc_type_top_field** and **chroma_sample_loc_type_bottom_field** syntax elements in the video usability information (VUI) parameters.

frame0_grid_position_x (when present) specifies the x component of the (x, y) coordinate pair for constituent frame 0.

frame0_grid_position_y (when present) specifies the y component of the (x, y) coordinate pair for constituent frame 0.

frame1_grid_position_x (when present) specifies the x component of the (x, y) coordinate pair for constituent frame 1.

frame1_grid_position_y (when present) specifies the y component of the (x, y) coordinate pair for constituent frame 1.

When **quincunx_sampling_flag** is equal to 0 and **frame_packing_arrangement_type** is not equal to 5 the (x, y) coordinate pair for each constituent frame is interpreted as follows:

- If the (x, y) coordinate pair for a constituent frame is equal to (0, 0), this indicates a default sampling grid alignment specified as follows:
 - If **frame_packing_arrangement_type** is equal to 3, the indicated position is the same as for the (x, y) coordinate pair value (4, 8), as illustrated in Figure D.4.
 - Otherwise (**frame_packing_arrangement_type** is equal to 4), the indicated position is the same as for the (x, y) coordinate pair value (8, 4), as illustrated in Figure D.6.
- Otherwise, if the (x, y) coordinate pair for a constituent frame is equal to (15, 15), this indicates that the sampling grid alignment is unknown or unspecified or specified by other means not specified in this Specification.
- Otherwise, the x and y elements of the (x, y) coordinate pair specify the indicated horizontal and vertical sampling grid alignment positioning to the right of and below the upper left corner of the rectangular area represented by the corresponding constituent frame, respectively, in units of one sixteenth of the luma sample grid spacing between the samples of the columns and rows of the constituent frame that are present in the decoded frame (prior to any upsampling for display or other purposes).

NOTE 8 – The spatial location reference information **frame0_grid_position_x**, **frame0_grid_position_y**, **frame1_grid_position_x**, and **frame1_grid_position_y** is not provided when **quincunx_sampling_flag** is equal to 1 because the spatial alignment in this case is assumed to be such that constituent frame 0 and constituent frame 1 cover corresponding spatial areas with interleaved quincunx sampling patterns as illustrated in Figure D.8.

frame_packing_arrangement_reserved_byte is reserved for future use by ITU-T | ISO/IEC. It is a requirement of bitstream conformance that the value of **frame_packing_arrangement_reserved_byte** shall be equal to 0. All other values of **frame_packing_arrangement_reserved_byte** are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of **frame_packing_arrangement_reserved_byte**.

frame_packing_arrangement_persistence_flag specifies the persistence of the frame packing arrangement SEI message for the current layer.

frame_packing_arrangement_persistence_flag equal to 0 specifies that the frame packing arrangement SEI message applies to the current decoded frame only.

Let **picA** be the current picture. **frame_packing_arrangement_persistence_flag** equal to 1 specifies that the frame packing arrangement SEI message persists for the current layer in output order until any of the following conditions are true:

- A new CLVS of the current layer begins.
- The bitstream ends.
- A frame **picB** in the current layer in an access unit containing a frame packing arrangement SEI message with the same value of **frame_packing_arrangement_id** and applicable to the current layer is output for which **PicOrderCnt(picB)** is greater than **PicOrderCnt(picA)**, where **PicOrderCnt(picB)** and **PicOrderCnt(picA)** are the **PicOrderCntVal** values of **picB** and **picA**, respectively, immediately after the invocation of the decoding process for picture order count for **picB**.

upsampled_aspect_ratio_flag equal to 1 indicates that the sample aspect ratio (SAR) indicated by the VUI parameters of the SPS identifies the SAR of the samples after the application of an upconversion process to produce a higher resolution

frame from each constituent frame as illustrated in Figure D.4 to Figure D.8. `upsampled_aspect_ratio_flag` equal to 0 indicates that the SAR indicated by the VUI parameters of the SPS identifies the SAR of the samples before the application of any such upconversion process.

NOTE 9 – The default display window parameters in the VUI parameters of the SPS can be used by an encoder to indicate to a decoder that does not interpret the frame packing arrangement SEI message that the default display window is an area within only one of the two constituent frames.

NOTE 10 – The SAR indicated in the VUI parameters should indicate the preferred display picture shape for the packed decoded frame output by a decoder that does not interpret the frame packing arrangement SEI message. When `upsampled_aspect_ratio_flag` is equal to 1, the SAR produced in each up-converted colour plane is indicated to be the same as the SAR indicated in the VUI parameters in the examples shown in Figure D.4 to Figure D.8. When `upsampled_aspect_ratio_flag` is equal to 0, the SAR produced in each colour plane prior to upconversion is indicated to be the same as the SAR indicated in the VUI parameters in the examples shown in Figure D.4 to Figure D.8.

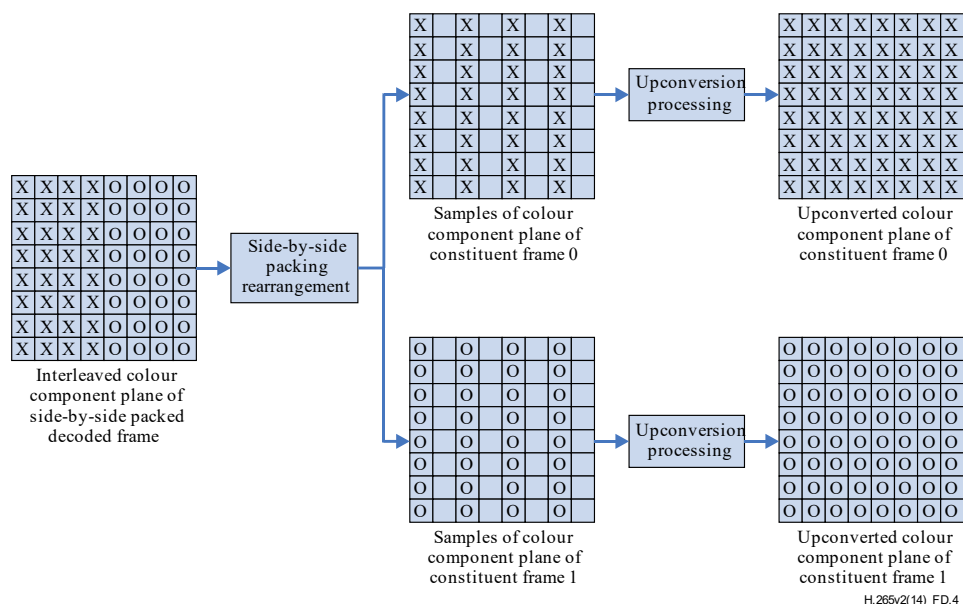


Figure D.4 – Rearrangement and upconversion of side-by-side packing arrangement with `frame_packing_arrangement_type` equal to 3, `quincunx_sampling_flag` equal to 0 and `(x, y)` equal to `(0, 0)` or `(4, 8)` for both constituent frames

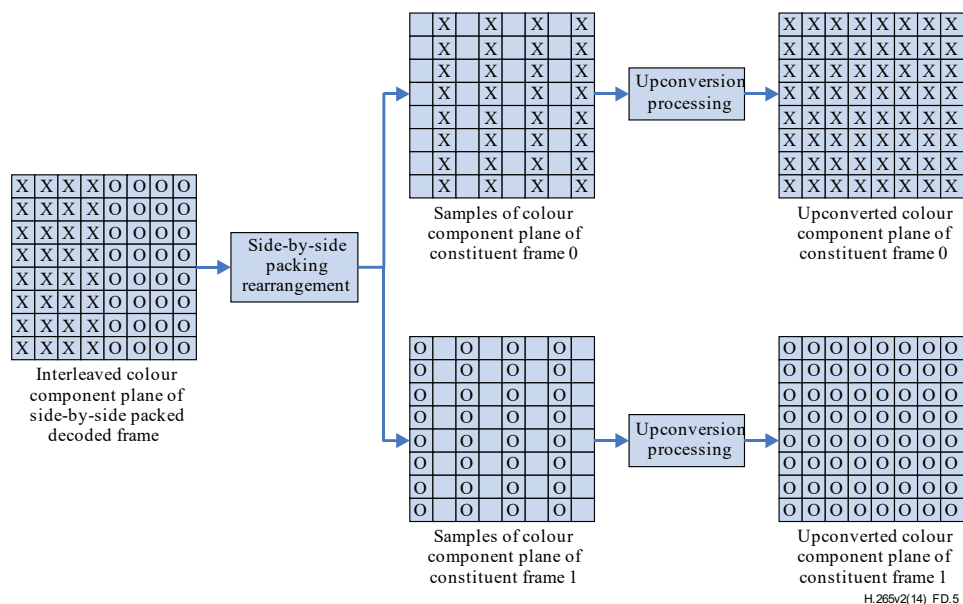


Figure D.5 – Rearrangement and upconversion of side-by-side packing arrangement with frame_packing_arrangement_type equal to 3, quincunx_sampling_flag equal to 0, (x, y) equal to (12, 8) for constituent frame 0 and (x, y) equal to (0, 0) or (4, 8) for constituent frame 1

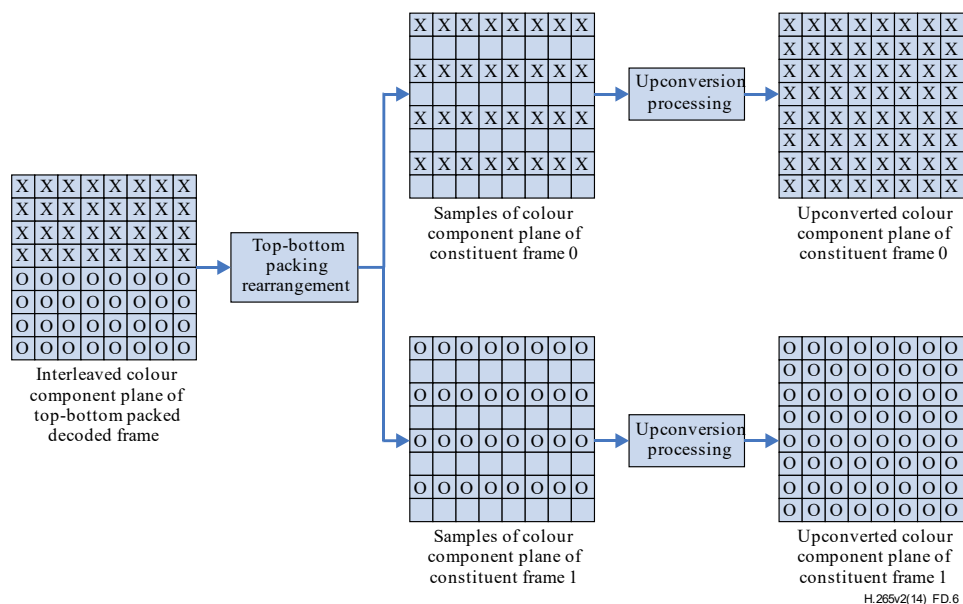


Figure D.6 – Rearrangement and upconversion of top-bottom packing arrangement with frame_packing_arrangement_type equal to 4, quincunx_sampling_flag equal to 0 and (x, y) equal to (0, 0) or (8, 4) for both constituent frames

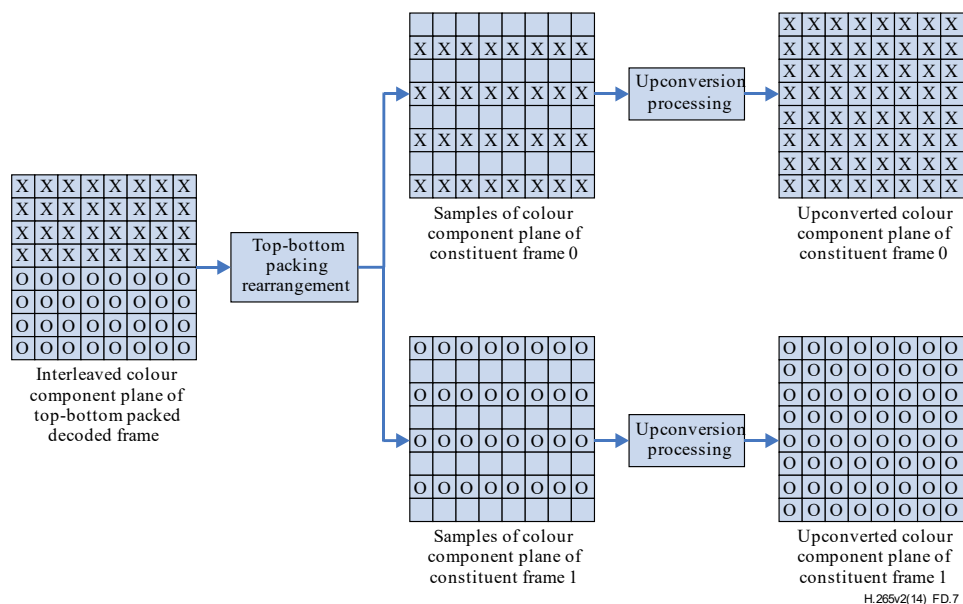


Figure D.7 – Rearrangement and upconversion of top-bottom packing arrangement with frame_packing_arrangement_type equal to 4, quincunx_sampling_flag equal to 0, (x, y) equal to (8, 12) for constituent frame 0 and (x, y) equal to (0, 0) or (8, 4) for constituent frame 1

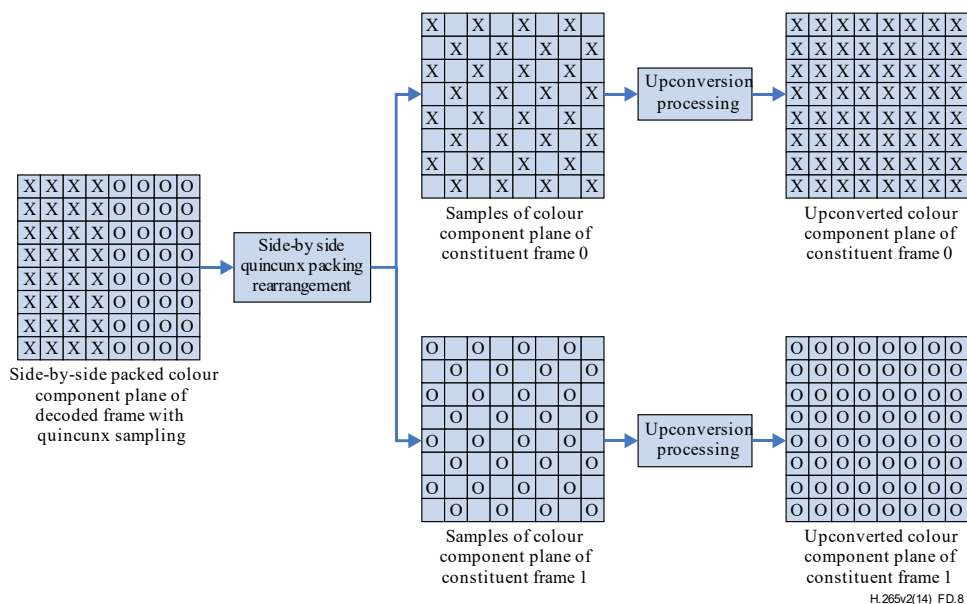
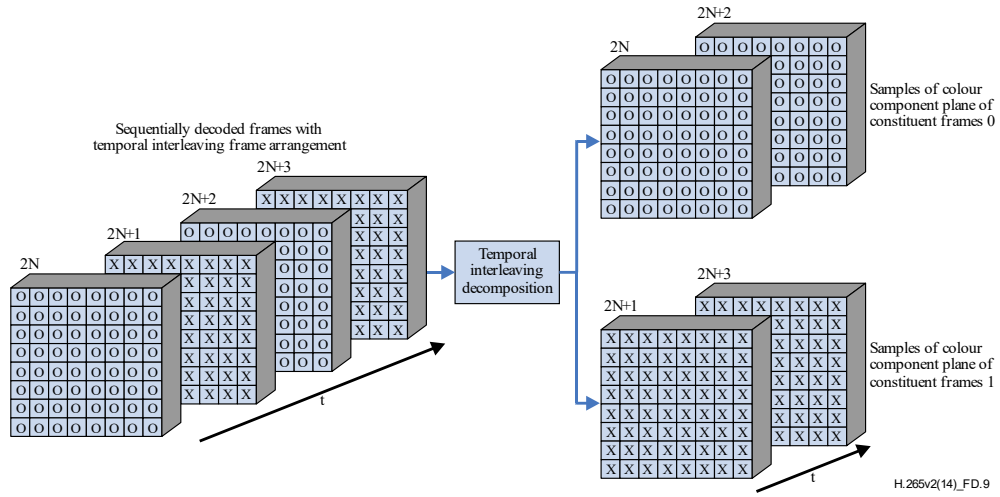


Figure D.8 – Rearrangement and upconversion of side-by-side packing arrangement with quincunx sampling (frame_packing_arrangement_type equal to 3 with quincunx_sampling_flag equal to 1)



**Figure D.9 – Rearrangement of a temporal interleaving frame arrangement
(frame_packing_arrangement_type equal to 5)**

D.3.17 Display orientation SEI message semantics

When the associated picture has PicOutputFlag equal to 1, the display orientation SEI message informs the decoder of a transformation that is recommended to be applied to the cropped decoded picture prior to display.

display_orientation_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous display orientation SEI message in output order. display_orientation_cancel_flag equal to 0 indicates that display orientation information follows.

hor_flip equal to 1 indicates that the cropped decoded picture should be flipped horizontally for display. hor_flip equal to 0 indicates that the decoded picture should not be flipped horizontally.

When hor_flip is equal to 1, the cropped decoded picture should be flipped as follows for each component Z equal to Y, Cb and Cr, letting dZ be the final cropped array of output samples for the component Z:

$$\begin{aligned} &\text{for}(x = 0; x < \text{croppedWidthZ}; x++) \\ &\quad \text{for}(y = 0; y < \text{croppedHeightZ}; y++) \\ &\quad \quad dZ[x][y] = Z[\text{croppedWidthZ} - x - 1][y] \end{aligned} \quad (\text{D-19})$$

Where croppedWidthZ is the width of the component Z of the cropped decoded picture in samples, croppedHeightZ is the height of the component Z of the cropped decoded picture in samples, and Z[x][y] and dZ[x][y] are the sample value before and after the horizontal flipping, respectively, for the sample at the location (x, y) of the component Z of the cropped decoded picture.

ver_flip equal to 1 indicates that the cropped decoded picture should be flipped vertically (in addition to any horizontal flipping when hor_flip is equal to 1) for display. ver_flip equal to 0 indicates that the decoded picture should not be flipped vertically.

When ver_flip is equal to 1, the cropped decoded picture should be flipped as follows for each component Z equal to Y, Cb and Cr, letting dZ be the final cropped array of output samples for the component Z:

$$\begin{aligned} &\text{for}(x = 0; x < \text{croppedWidthZ}; x++) \\ &\quad \text{for}(y = 0; y < \text{croppedHeightZ}; y++) \\ &\quad \quad dZ[x][y] = Z[x][\text{croppedWidthZ} - y - 1] \end{aligned} \quad (\text{D-20})$$

Where croppedWidthZ is the width of the component Z of the cropped decoded picture in samples, croppedHeightZ is the height of the component Z of the cropped decoded picture in samples, and Z[x][y] and dZ[x][y] are the sample value before and after the vertical flipping, respectively, for the sample at the location (x, y) of the component Z of the cropped decoded picture.

anticlockwise_rotation specifies the recommended anticlockwise rotation of the decoded picture (after applying horizontal or vertical flipping when hor_flip or ver_flip is set) prior to display. The decoded picture should be rotated by $360 * \text{anticlockwise_rotation} \div 2^{16}$ degrees ($2 * \pi * \text{anticlockwise_rotation} \div 2^{16}$ radians, where π is Archimedes' constant 3.141 592 653 589 793...) in the anticlockwise direction prior to display. For example, anticlockwise_rotation equal to 0

indicates no rotation and anticlockwise_rotation equal to 16 384 indicates 90 degrees ($\pi \div 2$ radians) rotation in the anticlockwise direction.

NOTE – It is possible for equivalent transformations to be expressed in multiple ways using these syntax elements. For example, the combination of having both hor_flip and ver_flip equal to 1 with anticlockwise_rotation equal to 0 can alternatively be expressed by having both hor_flip and ver_flip equal to 1 with anticlockwise_rotation equal to 0x8000, and the combination of hor_flip equal to 1 with ver_flip equal to 0 and anticlockwise_rotation equal to 0 can alternatively be expressed by having hor_flip equal to 0 with ver_flip equal to 1 and anticlockwise_rotation equal to 0x8000.

display_orientation_persistence_flag specifies the persistence of the display orientation SEI message for the current layer.

display_orientation_persistence_flag equal to 0 specifies that the display orientation SEI message applies to the current decoded picture only.

Let picA be the current picture. display_orientation_persistence_flag equal to 1 specifies that the display orientation SEI message persists for the current layer in output order until one or more of the following conditions are true:

- A new CLVS of the current layer begins.
- The bitstream ends.
- A picture picB in the current layer in an access unit containing a display orientation SEI message that is applicable to the current layer is output for which PicOrderCnt(picB) is greater than PicOrderCnt(picA), where PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

D.3.18 Green metadata SEI message semantics

The semantics for this SEI message are specified in ISO/IEC 23001-11 (Green metadata). Green metadata facilitates reduced power consumption in decoders, encoders, displays and in media selection.

D.3.19 Structure of pictures information SEI message semantics

The structure of pictures information SEI message provides information for a list of entries, some of which correspond to the target picture set that consists of a series of pictures starting from the current picture until the last picture in decoding order in the current layer in the CLVS.

The first entry in the structure of pictures information SEI message corresponds to the current picture. When there is a picture in the target picture set that has PicOrderCntVal equal to the variable entryPicOrderCnt[i] as specified below, the entry i corresponds to a picture in the target picture set. The decoding order of the pictures in the target picture set that correspond to entries in the structure of pictures information SEI message corresponds to increasing values of i in the list of entries.

Any picture picB in the target picture set that has PicOrderCntVal equal to entryPicOrderCnt[i] for any i in the range of 0 to num_entries_in_sop_minus1, inclusive, where PicOrderCntVal is the value of PicOrderCntVal of picB immediately after the invocation of the decoding process for picture order count for picB, shall correspond to an entry in the list of entries.

The structure of pictures information SEI message shall not be present in a CVS and applicable for a layer for which the active SPS has long_term_ref_pics_present_flag equal to 1 or num_short_term_ref_pic_sets equal to 0.

The structure of pictures information SEI message shall not be present in any access unit that has TemporalId greater than 0 or contains a RASL, RADL or SLNR picture in the current layer. Any picture in the target picture set that corresponds to an entry other than the first entry described in the structure of pictures information SEI message shall not be an IRAP picture.

sop_seq_parameter_set_id indicates and shall be equal to the sps_seq_parameter_set_id value of the active SPS. The value of sop_seq_parameter_set_id shall be in the range of 0 to 15, inclusive.

num_entries_in_sop_minus1 plus 1 specifies the number of entries in the structure of pictures information SEI message. num_entries_in_sop_minus1 shall be in the range of 0 to 1 023, inclusive.

sop_vcl_nut[i], when the i-th entry corresponds to a picture in the target picture set, indicates and shall be equal to the nal_unit_type value of the picture corresponding to the i-th entry.

sop_temporal_id[i], when the i-th entry corresponds to a picture in the target picture set, indicates and shall be equal to the TemporalId value of the picture corresponding to the i-th entry. The value of 7 for sop_temporal_id[i] is reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore structure of pictures information SEI messages that contain the value 7 for sop_temporal_id[i].

sop_short_term_rps_idx[i], when the i-th entry corresponds to a picture in the target picture set, indicates and shall be equal to the index, into the list of candidate short-term RPSs included in the active SPS, of the candidate short-term RPS used by the picture corresponding to the i-th entry for derivation of the short-term reference picture set. **sop_short_term_rps_idx**[i] shall be in the range of 0 to **num_short_term_ref_pic_sets** – 1, inclusive.

sop_poc_delta[i] is used to specify the value of the variable **entryPicOrderCnt**[i] for the i-th entry described in the structure of pictures information SEI message. **sop_poc_delta**[i] shall be in the range of (–**MaxPicOrderCntLsb**) / 2 + 1 to **MaxPicOrderCntLsb** / 2 – 1, inclusive.

The variable **entryPicOrderCnt**[i] is derived as follows:

$$\begin{aligned} \text{entryPicOrderCnt}[0] &= \text{PicOrderCnt}(\text{currPic}) \\ \text{for}(i = 1; i \leq \text{num_entries_in_sop_minus1}; i++) \\ \quad \text{entryPicOrderCnt}[i] &= \text{entryPicOrderCnt}[i - 1] + \text{sop_poc_delta}[i] \end{aligned} \quad (\text{D-21})$$

where **currPic** is the current picture.

D.3.20 Decoded picture hash SEI message semantics

This message provides a hash for each colour component of the current decoded picture.

NOTE 1 – The decoded picture hash SEI message is a suffix SEI message and cannot be contained in a scalable nesting SEI message.

Prior to computing the hash, the decoded picture data are arranged into one or three strings of bytes called **pictureData**[cIdx] of lengths **dataLen**[cIdx] as follows:

$$\begin{aligned} &\text{for}(cIdx = 0; cIdx < (\text{chroma_format_idc} == 0) ? 1 : 3; cIdx++) \{ \\ &\quad \text{if}(cIdx == 0) \{ \\ &\quad \quad \text{compWidth}[cIdx] = \text{pic_width_in_luma_samples} \\ &\quad \quad \text{compHeight}[cIdx] = \text{pic_height_in_luma_samples} \\ &\quad \quad \text{compDepth}[cIdx] = \text{BitDepth}_Y \\ &\quad \} \text{ else } \{ \\ &\quad \quad \text{compWidth}[cIdx] = \text{pic_width_in_luma_samples} / \text{SubWidthC} \\ &\quad \quad \text{compHeight}[cIdx] = \text{pic_height_in_luma_samples} / \text{SubHeightC} \\ &\quad \quad \text{compDepth}[cIdx] = \text{BitDepth}_C \\ &\quad \} \\ &\quad iLen = 0 \\ &\quad \text{for}(i = 0; i < \text{compWidth}[cIdx] * \text{compHeight}[cIdx]; i++) \{ \\ &\quad \quad \text{pictureData}[cIdx][iLen++] = \text{component}[cIdx][i] \& 0xFF \\ &\quad \quad \text{if}(\text{compDepth}[cIdx] > 8) \\ &\quad \quad \quad \text{pictureData}[cIdx][iLen++] = \text{component}[cIdx][i] >> 8 \\ &\quad \} \\ &\quad \text{dataLen}[cIdx] = iLen \\ &\} \end{aligned} \quad (\text{D-22})$$

where **component**[cIdx][i] is an array in raster scan of decoded sample values in two's complement representation.

hash_type indicates the method used to calculate the checksum according to Table D.10. Values of **hash_type** that are not listed in Table D.10 are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore decoded picture hash SEI messages that contain reserved values of **hash_type**.

Table D.10 – Interpretation of hash_type

hash_type	Method
0	MD5 (IETF RFC 1321)
1	CRC
2	Checksum

picture_md5[cIdx][i] is the 16-byte MD5 hash of the cIdx-th colour component of the decoded picture. The value of **picture_md5**[cIdx][i] shall be equal to the value of **digestVal**[cIdx] obtained as follows, using the MD5 functions defined in IETF RFC 1321:

```

MD5Init( context )
MD5Update( context, pictureData[ cIdx ], dataLen[ cIdx ] )
MD5Final( digestVal[ cIdx ], context )

```

(D-23)

picture_crc[cIdx] is the cyclic redundancy check (CRC) of the colour component cIdx of the decoded picture. The value of picture_crc[cIdx] shall be equal to the value of crcVal[cIdx] obtained as follows:

```

crc = 0xFFFF
pictureData[ cIdx ][ dataLen[ cIdx ] ] = 0
pictureData[ cIdx ][ dataLen[ cIdx ] + 1 ] = 0
for( bitIdx = 0; bitIdx < ( dataLen[ cIdx ] + 2 ) * 8; bitIdx++ ) {
    dataByte = pictureData[ cIdx ][ bitIdx >> 3 ]
    crcMsb = ( crc >> 15 ) & 1
    bitVal = ( dataByte >> ( 7 - ( bitIdx & 7 ) ) ) & 1
    crc = ( ( ( crc << 1 ) + bitVal ) & 0xFFFF ) ^ ( crcMsb * 0x1021 )
}
crcVal[ cIdx ] = crc

```

(D-24)

NOTE 2 – The same CRC specification is found in Rec. ITU-T H.271.

picture_checksum[cIdx] is the checksum of the colour component cIdx of the decoded picture. The value of picture_checksum[cIdx] shall be equal to the value of checksumVal[cIdx] obtained as follows:

```

sum = 0
for( y = 0; y < compHeight[ cIdx ]; y++ )
    for( x = 0; x < compWidth[ cIdx ]; x++ ) {
        xorMask = ( x & 0xFF ) ^ ( y & 0xFF ) ^ ( x >> 8 ) ^ ( y >> 8 )
        sum = ( sum + ( ( component[ cIdx ][ y * compWidth[ cIdx ] + x ] & 0xFF ) ^ xorMask ) )
        &
        0xFFFFFFFF
        if( compDepth[ cIdx ] > 8 )
            sum = ( sum + ( ( component[ cIdx ][ y * compWidth[ cIdx ] + x ] >> 8 ) ^ xorMask ) )
    } &
    0xFFFFFFFF
checksumVal[ cIdx ] = sum

```

(D-25)

D.3.21 Active parameter sets SEI message semantics

The active parameter sets SEI message indicates which VPS is active for the VCL NAL units of the access unit associated with the SEI message. The SEI message may also provide information on which SPS is active for the VCL NAL units of the access unit associated with the SEI message and other information related to parameter sets.

active_video_parameter_set_id indicates and shall be equal to the value of the vps_video_parameter_set_id of the VPS that is referred to by the VCL NAL units of the access unit associated with the SEI message. The value of active_video_parameter_set_id shall be in the range of 0 to 15, inclusive.

self_contained_cvs_flag equal to 1 indicates that each parameter set that is (directly or indirectly) referenced by any VCL NAL unit of the CVS that is not a VCL NAL unit of a RASL picture is present within the CVS at a position that precedes, in decoding order, any NAL unit that (directly or indirectly) references the parameter set. self_contained_cvs_flag equal to 0 indicates that this property may or may not apply.

no_parameter_set_update_flag equal to 1 indicates that there is no parameter set update in the CVS, i.e., each VPS in the CVS is an exact copy of the previous VPS in decoding order in the bitstream that has the same value of vps_video_parameter_set_id, when present; each SPS in the CVS is an exact copy of the previous SPS in decoding order in the bitstream that has the same value of sps_seq_parameter_set_id, when present; and each PPS in the CVS is an exact copy of the previous PPS in decoding order in the bitstream that has the same value of pps_pic_parameter_set_id, when present. no_parameter_set_update_flag equal to 0 indicates that there may or may not be parameter set update in the CVS.

NOTE 1 – If no_parameter_set_update_flag equal to 1 is indicated for each CVS in a bitstream, i.e., there is no parameter set update in the bitstream, it is possible to transmit all parameter sets out-of-band before sending the first VCL NAL unit of the bitstream or to place all parameter sets at the beginning of the bitstream. Otherwise, out-of-band transmission of all parameter sets before sending VCL NAL units is not possible.

num_sps_ids_minus1 plus 1 shall be less than or equal to the number of SPSs that are referred to by the VCL NAL units of the access unit associated with the active parameter sets SEI message. The value of **num_sps_ids_minus1** shall be in the range of 0 to 15, inclusive.

active_seq_parameter_set_id[*i*] indicates a value of the **sps_seq_parameter_set_id** of the SPS that may be referred to by any VCL NAL unit of the access unit associated with the SEI message. When **vps_base_layer_internal_flag** is equal to 1 and **vps_base_layer_available_flag** is equal to 1, **active_seq_parameter_set_id**[0] indicates and shall be equal to the value of the **sps_seq_parameter_set_id** of the SPS that is referred to by the VCL NAL units with **nuh_layer_id** equal to 0 in the access unit associated with the SEI message. The value of **active_seq_parameter_set_id**[*i*] shall be in the range of 0 to 15, inclusive.

layer_sps_idx[*i*] indicates that the **sps_seq_parameter_set_id** value of the SPS that is referred to by the VCL NAL units with **nuh_layer_id** equal to **layer_id_in_nuh**[*i*] in the access unit associated with the SEI message, if any, is equal to **active_seq_parameter_set_id**[**layer_sps_idx**[*i*]].

NOTE 2 – When decoding a single-layer bitstream or only the base layer of a multi-layer bitstream, the decoder does not need to parse **layer_sps_idx**[*i*] syntax elements, because they do not affect the decoding of the base layer.

D.3.22 Decoding unit information SEI message semantics

The decoding unit information SEI message provides CPB removal delay information for the decoding unit associated with the SEI message.

When the buffering period SEI message is non-scalable-nested or is directly contained in a scalable nesting SEI message within an SEI NAL unit with **nuh_layer_id** equal to 0, the following applies for the decoding unit information SEI message syntax and semantics:

- The syntax elements **sub_pic_hrd_params_present_flag**, **sub_pic_cpb_params_in_pic_timing_sei_flag** and **dpb_output_delay_du_length_minus1**, and the variable **CpbDpbDelaysPresentFlag** are found in or derived from syntax elements in the **hrd_parameters()** syntax structure that is applicable to at least one of the operation points to which the decoding unit information SEI message applies.
- The bitstream (or a part thereof) refers to the bitstream subset (or a part thereof) associated with any of the operation points to which the decoding unit information SEI message applies.

The presence of decoding unit information SEI messages for an operation point including the base layer is specified as follows:

- If **CpbDpbDelaysPresentFlag** is equal to 1, **sub_pic_hrd_params_present_flag** is equal to 1 and **sub_pic_cpb_params_in_pic_timing_sei_flag** is equal to 0, one or more decoding unit information SEI messages applicable to the operation point shall be associated with each decoding unit in the CVS.
- Otherwise, if **CpbDpbDelaysPresentFlag** is equal to 1, **sub_pic_hrd_params_present_flag** is equal to 1 and **sub_pic_cpb_params_in_pic_timing_sei_flag** is equal to 1, one or more decoding unit information SEI messages applicable to the operation point may or may not be associated with each decoding unit in the CVS.
- Otherwise (**CpbDpbDelaysPresentFlag** is equal to 0 or **sub_pic_hrd_params_present_flag** is equal to 0), in the CVS there shall be no decoding unit that is associated with a decoding unit information SEI message applicable to the operation point.

When the buffering period SEI message is non-scalable-nested or is directly contained in a scalable nesting SEI message within an SEI NAL unit with **nuh_layer_id** equal to 0, the set of NAL units associated with a decoding unit information SEI message consists, in decoding order, of the SEI NAL unit containing the decoding unit information SEI message and all subsequent NAL units in the access unit up to but not including any subsequent SEI NAL unit containing a decoding unit information SEI message with a different value of **decoding_unit_idx**. Each decoding unit shall include at least one VCL NAL unit. All non-VCL NAL units associated with a VCL NAL unit shall be included in the decoding unit containing the VCL NAL unit.

decoding_unit_idx specifies the index, starting from 0, to the list of decoding units in the current access unit, of the decoding unit associated with the decoding unit information SEI message. The value of **decoding_unit_idx** shall be in the range of 0 to **PicSizeInCtbsY** – 1, inclusive.

A decoding unit identified by a particular value of **duIdx** includes and only includes all NAL units associated with all decoding unit information SEI messages that have **decoding_unit_idx** equal to **duIdx**. Such a decoding unit is also referred to as associated with the decoding unit information SEI messages having **decoding_unit_idx** equal to **duIdx**.

For any two decoding units **duA** and **duB** in one access unit with **decoding_unit_idx** equal to **duIdxA** and **duIdxB**, respectively, where **duIdxA** is less than **duIdxB**, **duA** shall precede **duB** in decoding order.

A NAL unit of one decoding unit shall not be present, in decoding order, between any two NAL units of another decoding unit.

du_spt_cpb_removal_delay_increment specifies the duration, in units of clock sub-ticks, between the nominal CPB times of the last decoding unit in decoding order in the current access unit and the decoding unit associated with the decoding unit information SEI message. This value is also used to calculate an earliest possible time of arrival of decoding unit data into the CPB for the HSS, as specified in Annex C or clause F.13. The syntax element is represented by a fixed length code whose length in bits is given by `du_cpb_removal_delay_increment_length_minus1 + 1`. When the decoding unit associated with the decoding unit information SEI message is the last decoding unit in the current access unit, the value of `du_spt_cpb_removal_delay_increment` shall be equal to 0.

dpb_output_du_delay_present_flag equal to 1 specifies the presence of the `pic_spt_dpb_output_du_delay` syntax element in the decoding unit information SEI message. `dpb_output_du_delay_present_flag` equal to 0 specifies the absence of the `pic_spt_dpb_output_du_delay` syntax element in the decoding unit information SEI message.

pic_spt_dpb_output_du_delay is used to compute the DPB output time of the picture when `SubPicHrdFlag` is equal to 1. It specifies how many sub clock ticks to wait after removal of the last decoding unit in an access unit from the CPB before the decoded picture is output from the DPB. When not present, the value of `pic_spt_dpb_output_du_delay` is inferred to be equal to `pic_dpb_output_du_delay`.

The length of the syntax element `pic_spt_dpb_output_du_delay` is given in bits by `dpb_output_delay_du_length_minus1 + 1`.

When the buffering period SEI message is non-scalable-nested or is directly contained in a scalable nesting SEI message within an SEI NAL unit with `nuh_layer_id` equal to 0, it is a requirement of bitstream conformance that all decoding unit information SEI messages that are associated with the same access unit, apply to the same operation point, and have `dpb_output_du_delay_present_flag` equal to 1 shall have the same value of `pic_spt_dpb_output_du_delay`.

The output time derived from the `pic_spt_dpb_output_du_delay` of any picture that is output from an output timing conforming decoder shall precede the output time derived from the `pic_spt_dpb_output_du_delay` of all pictures in any subsequent CVS in decoding order.

The picture output order established by the values of this syntax element shall be the same order as established by the values of `PicOrderCntVal`.

For pictures that are not output by the "bumping" process because they precede, in decoding order, an IRAP picture with `NoRaslOutputFlag` equal to 1 that has `no_output_of_prior_pics_flag` equal to 1 or inferred to be equal to 1, the output times derived from `pic_spt_dpb_output_du_delay` shall be increasing with increasing value of `PicOrderCntVal` relative to all pictures within the same CVS.

For any two pictures in the CVS, the difference between the output times of the two pictures when `SubPicHrdFlag` is equal to 1 shall be identical to the same difference when `SubPicHrdFlag` is equal to 0.

D.3.23 Temporal sub-layer zero index SEI message semantics

The temporal sub-layer zero index SEI message provides information that can assist the decoder for detection of missing coded pictures that have `TemporalId` equal to 0 and are not RASL, RADL or SLNR pictures.

When a temporal sub-layer zero index SEI message is present in the current access unit and applies to the current layer and the current picture is not an IRAP picture, a temporal sub-layer zero index SEI message that applies to the current layer shall also be present in the preceding access unit in decoding order with `TemporalId` equal to 0 and containing a picture that is not a RASL, RADL or SLNR picture in the current layer.

NOTE – Encoders should be cautious when setting `discardable_flag` equal to 1 for any picture with `TemporalId` equal to 0, as based on temporal sub-layer zero index SEI messages decoders may consider intentionally discarded pictures with `TemporalId` equal to 0 and `discardable_flag` equal to 1 as lost and take unnecessary error recovery actions, such as requesting retransmission of a missing picture with `TemporalId` equal to 0.

Let `ReferencedTsl0Pic` denote a picture that has `TemporalId` equal to 0 and is not a RASL, RADL or SLNR picture.

temporal_sub_layer_zero_idx indicates a temporal sub-layer zero index as follows:

- If the current picture is a `ReferencedTsl0Pic`, `temporal_sub_layer_zero_idx` indicates the temporal sub-layer zero index for the current picture.
- Otherwise, `temporal_sub_layer_zero_idx` indicates the temporal sub-layer zero index of the preceding picture in the current layer in decoding order that is a `ReferencedTsl0Pic`.

When the bitstream contains a preceding access unit in decoding order that contained a `ReferencedTsl0Pic`, and that preceding access unit has an associated temporal sub-layer zero index SEI message `seiMsgB` that applies to the current layer, the variable `prevTsl0Idx` is set equal to the value of `temporal_sub_layer_zero_idx` of `seiMsgB`.

The following constraints apply to the value of `temporal_sub_layer_zero_idx`:

- If the current picture is an IRAP picture, `temporal_sub_layer_zero_idx` shall be equal to 0.

- Otherwise, the following applies:
 - If the current picture is a ReferencedTsl0Pic, temporal_sub_layer_zero_idx shall be equal to $(\text{prevTsl0Idx} + 1) \% 256$.
 - Otherwise, temporal_sub_layer_zero_idx shall be equal to prevTsl0Idx.

irap_pic_id is an IRAP picture identifier for the current layer. When the current picture is not the first picture in the current layer in the bitstream in decoding order and the preceding IRAP picture in the current layer in decoding order has an associated temporal sub-layer zero index SEI message, the following constraints apply to the value of irap_pic_id:

- If the current picture is an IRAP picture, irap_pic_id shall differ in value from the value of irap_pic_id of the temporal sub-layer zero index SEI message of the preceding IRAP picture in the current layer in decoding order.

NOTE – It is suggested for the value of irap_pic_id to be set to a random value (subject to the constraints specified herein), to minimize the likelihood of duplicate values appearing in a layer in the bitstream due to picture losses or splicing operations.
- Otherwise, irap_pic_id shall be equal to the value of irap_pic_id of the temporal sub-layer zero index SEI message associated with the preceding IRAP picture in the current layer in decoding order.

D.3.24 Scalable nesting SEI message semantics

The scalable nesting SEI message provides a mechanism to associate SEI messages with bitstream subsets corresponding to various operation points or with specific layers or sub-layers.

A scalable nesting SEI message contains one or more SEI messages.

NOTE – A scalable nesting SEI message can only be the last SEI message in the SEI NAL unit containing the scalable nesting SEI message. The sei_payload() of the scalable nesting SEI message cannot contain any reserved_payload_extension_data.

It is a requirement of bitstream conformance that the following restrictions apply on containing of SEI messages in a scalable nesting SEI message:

- An SEI message that has payloadType equal to 129 (active parameter sets), 132 (decoded picture hash) or 133 (scalable nesting) shall not be contained in a scalable nesting SEI message.
- When a scalable nesting SEI message contains a buffering period SEI message, a picture timing SEI message or a decoding unit information SEI message, the scalable nesting SEI message shall not contain any other SEI message with payloadType not equal to 0 (buffering period), 1 (picture timing) or 130 (decoding unit information).

bitstream_subset_flag equal to 0 specifies that the SEI messages contained in the scalable nesting SEI message apply to specific layers or sub-layers. bitstream_subset_flag equal to 1 specifies that the SEI messages contained in the scalable nesting SEI message apply to one or more sub-bitstreams as specified below.

Depending on the value of bitstream_subset_flag, the layers or sub-layers, or the operation points to which the SEI messages contained in the scalable nesting SEI message apply are specified by deriving the lists nestingLayerIdList[i] and the variables MaxTemporalId[i] based on syntax element values as specified below.

It is a requirement of bitstream conformance that the following restrictions apply on the value of bitstream_subset_flag:

- When the scalable nesting SEI message contains a buffering period SEI message, a picture timing SEI message or a decoding unit information SEI message, bitstream_subset_flag shall be equal to 1.
- When the scalable nesting SEI message contains an SEI message that has payloadType equal to any value among SingleLayerSeiList, bitstream_subset_flag shall be equal to 0.

nesting_op_flag equal to 0 specifies that the list nestingLayerIdList[0] is specified by all_layers_flag and, when present, nesting_layer_id[i] for all i values in the range of 0 to nesting_num_layers_minus1, inclusive, and that the variable MaxTemporalId[0] is specified by nesting_no_op_max_temporal_id_plus1. nesting_op_flag equal to 1 specifies that the list nestingLayerIdList[i] and the variable MaxTemporalId[i] are specified by nesting_num_ops_minus1, default_op_flag, nesting_max_temporal_id_plus1[i], when present, and nesting_op_idx[i], when present.

default_op_flag equal to 1 specifies that MaxTemporalId[0] is equal to nuh_temporal_id_plus1 of the current SEI NAL unit minus 1 and that nestingLayerIdList[0] contains all integer values in the range of 0 to nuh_layer_id of the current SEI NAL unit, inclusive, in increasing order of the values.

nesting_num_ops_minus1 plus 1 minus default_op_flag specifies the number of the following nesting_op_idx[i] syntax elements. The value of nesting_num_ops_minus1 shall be in the range of 0 to 1 023, inclusive.

If nesting_op_flag is equal to 0, the variable nestingNumOps is set equal to 1. Otherwise, the variable nestingNumOps is set equal to nesting_num_ops_minus1 + 1.

nesting_max_temporal_id_plus1[*i*] is used to specify the variable **MaxTemporalId**[*i*]. The value of **nesting_max_temporal_id_plus1**[*i*] shall be greater than or equal to **nuh_temporal_id_plus1** of the current SEI NAL unit. The variable **MaxTemporalId**[*i*] is set equal to **nesting_max_temporal_id_plus1**[*i*] – 1.

nesting_op_idx[*i*] is used to specify the list **nestingLayerIdList**[*i*]. The value of **nesting_op_idx**[*i*] shall be in the range of 0 to 1 023, inclusive.

The list **nestingLayerIdList**[*i*] is set equal to the **OpLayerIdList** of the **nesting_op_idx**[*i*]-th layer set specified by the active VPS.

It is a requirement of bitstream conformance that when **nesting_op_flag** is equal to 1 and **nesting_op_idx**[*i*] is greater than **vps_num_layer_sets_minus1** for any value of *i* in the range of **default_op_flag** to **nesting_num_ops_minus1**, inclusive, the value of **nuh_layer_id** of the SEI NAL unit containing the scalable nesting SEI message shall be greater than 0.

all_layers_flag equal to 0 specifies that the list **nestingLayerIdList**[0] is specified by **nesting_layer_id**[*i*] for all *i* values in the range of 0 to **nesting_num_layers_minus1**, inclusive. **all_layers_flag** equal to 1 specifies that the list **nestingLayerIdList**[0] consists of all values of **nuh_layer_id** present in the current access unit that are greater than or equal to **nuh_layer_id** of the current SEI NAL unit, in increasing order of the values.

When **nesting_op_flag** is equal to 0 and **all_layers_flag** is equal to 1, **MaxTemporalId**[0] is set equal to 6.

nesting_no_op_max_temporal_id_plus1 minus 1 specifies the value of **MaxTemporalId**[0] when **nesting_op_flag** is equal to 0 and **all_layers_flag** is equal to 0. The value of **nesting_no_op_max_temporal_id_plus1** shall not be equal to 0.

nesting_num_layers_minus1 plus 1 specifies the number of the following **nesting_layer_id**[*i*] syntax elements. The value of **nesting_num_layers_minus1** shall be in the range of 0 to 63, inclusive.

nesting_layer_id[*i*] specifies the *i*-th **nuh_layer_id** value included in the list **nestingLayerIdList**[0].

For any *i* and *j* in the range of 0 to **nesting_num_layers_minus1**, inclusive, with *i* less than *j*, **nesting_layer_id**[*i*] shall be less than **nesting_layer_id**[*j*].

The list **nestingLayerIdList**[0] is set to consist of **nesting_layer_id**[*i*] for all *i* values in the range of 0 to **nesting_num_layers_minus1**, inclusive, in increasing order of *i* values.

When **bitstream_subset_flag** is equal to 0, the following applies:

- The SEI messages contained in the scalable nesting SEI message apply to the sets of layers or sub-layers **subLayerSet**[*i*] for all *i* values in the range of 0 to **nestingNumOps** – 1, inclusive, where the VCL NAL units of the layers or sub-layers in each set **subLayerSet**[*i*] have **nuh_layer_id** values that are included in the list **nestingLayerIdList**[*i*] and **TemporalId** values that are in the range of the **TemporalId** of the current SEI NAL unit to **MaxTemporalId**[*i*], inclusive.
- When a scalable-nested SEI message has **payloadType** equal to any value among **VclAssociatedSeiList**, the value of **TemporalId** of the SEI NAL unit containing the scalable nesting SEI message shall be equal to 0 and **MaxTemporalId**[*i*] for all values of *i* in the range of 0 to **nestingNumOps** – 1, inclusive, shall be equal to 6.
- When a scalable-nested SEI message has **payloadType** equal to any value among **VclAssociatedSeiList** and the value of **nestingNumOps** is greater than 0, the scalable-nested SEI message applies to all layers for which each **nuh_layer_id** is included in at least one of the lists **nestingLayerIdList**[*i*] with *i* ranging from 0 to **nestingNumOps** – 1, inclusive.

When **bitstream_subset_flag** is equal to 1, the SEI messages contained in the scalable nesting SEI message apply to the operation points corresponding to the sub-bitstreams **subBitstream**[*i*] for all *i* values in the range of 0 to **nestingNumOps** – 1, inclusive, where each sub-bitstream **subBitstream**[*i*] is derived as follows:

- If **nestingLayerIdList**[*i*][0] is equal to 0 and **vps_base_layer_internal_flag** is equal to 1, the sub-bitstream **subBitstream**[*i*] is the output of the sub-bitstream extraction process of clause 10 with the bitstream, **MaxTemporalId**[*i*] and **nestingLayerIdList**[*i*] as inputs.
- Otherwise, if **nestingLayerIdList**[*i*][0] is equal to 0 and **vps_base_layer_internal_flag** is equal to 0, the sub-bitstream **subBitstream**[*i*] is the output of the sub-bitstream extraction process of clause F.10.1 with the bitstream, **MaxTemporalId**[*i*] and **nestingLayerIdList**[*i*] as inputs.
- Otherwise, the sub-bitstream **subBitstream**[*i*] is the output of the sub-bitstream extraction process of clause F.10.3 with the bitstream, **MaxTemporalId**[*i*] and **nestingLayerIdList**[*i*] as inputs.

nesting_zero_bit shall be equal to 0.

D.3.25 Region refresh information SEI message semantics

The region refresh information SEI message indicates whether the slice segments that the current SEI message applies to belong to a refreshed region of the current picture (as defined below).

An access unit that is not an IRAP access unit and that contains a recovery point SEI message is referred to as a gradual decoding refresh (GDR) access unit, and its corresponding picture is referred to as a GDR picture. The access unit corresponding to the indicated recovery point picture is referred to as the recovery point access unit.

If there is a picture that follows the GDR picture in decoding order in the CVS and that has `PicOrderCntVal` equal to the `PicOrderCntVal` of the GDR picture plus the value of `recovery_poc_cnt` in the recovery point SEI message, let the variable `lastPicInSet` be the recovery point picture. Otherwise, let `lastPicInSet` be the picture that immediately precedes the recovery point picture in output order. The picture `lastPicInSet` shall not precede the GDR picture in decoding order.

Let `gdrPicSet` be the set of pictures starting from a GDR picture to the picture `lastPicInSet`, inclusive, in output order. When the decoding process is started from a GDR access unit, the refreshed region in each picture of the `gdrPicSet` is indicated to be the region of the picture that is correct or approximately correct in content, and, when `lastPicInSet` is the recovery point picture, the refreshed region in `lastPicInSet` covers the entire picture.

The slice segments to which a region refresh information SEI message applies consist of all slice segments within the access unit that follow the SEI NAL unit containing the region refresh information SEI message and precede the next SEI NAL unit containing a region refresh information SEI message (if any) in decoding order. These slice segments are referred to as the slice segments associated with the region refresh information SEI message.

Let `gdrAuSet` be the set of access units corresponding to `gdrPicSet`. A `gdrAuSet` and the corresponding `gdrPicSet` are referred to as being associated with the recovery point SEI message contained in the GDR access unit.

Region refresh information SEI messages shall not be present in an access unit unless the access unit is included in a `gdrAuSet` associated with a recovery point SEI message. When any access unit that is included in a `gdrAuSet` contains one or more region refresh information SEI messages, all access units in the `gdrAuSet` shall contain one or more region refresh information SEI messages.

refreshed_region_flag equal to 1 indicates that the slice segments associated with the current SEI message belong to the refreshed region in the current picture. **refreshed_region_flag** equal to 0 indicates that the slice segments associated with the current SEI message may not belong to the refreshed region in the current picture.

When one or more region refresh information SEI messages are present in an access unit and the first slice segment of the access unit in decoding order does not have an associated region refresh information SEI message, the value of **refreshed_region_flag** for the slice segments that precede the first region refresh information SEI message is inferred to be equal to 0.

When `lastPicInSet` is the recovery point picture, and any region refresh SEI message is included in a recovery point access unit, the first slice segment of the access unit in decoding order shall have an associated region refresh SEI message, and the value of **refreshed_region_flag** shall be equal to 1 in all region refresh SEI messages in the access unit.

When one or more region refresh information SEI messages are present in an access unit, the refreshed region in the picture is specified as the set of CTUs in all slice segments of the access unit that are associated with region refresh information SEI messages that have **refreshed_region_flag** equal to 1. Other slice segments belong to the non-refreshed region of the picture.

It is a requirement of bitstream conformance that when a dependent slice segment belongs to the refreshed region, the preceding slice segment in decoding order shall also belong to the refreshed region.

Let `gdrRefreshedSliceSegmentSet` be the set of all slice segments that belong to the refreshed regions in the `gdrPicSet`. When a `gdrAuSet` contains one or more region refresh information SEI messages, it is a requirement of bitstream conformance that the following constraints all apply:

- The refreshed region in the first picture included in the corresponding `gdrPicSet` in decoding order that contains any refreshed region shall contain only coding units that are coded in an intra coding mode.
- For each picture included in the `gdrPicSet`, the syntax elements in `gdrRefreshedSliceSegmentSet` shall be constrained such that no samples or motion vector values outside of `gdrRefreshedSliceSegmentSet` are used for inter prediction in the decoding process of any samples within `gdrRefreshedSliceSegmentSet`.
- For any picture that follows the picture `lastPicInSet` in output order, the syntax elements in the slice segments of the picture shall be constrained such that no samples or motion vector values outside of `gdrRefreshedSliceSegmentSet` are used for inter prediction in the decoding process of the picture other than those of the other pictures that follow the picture `lastPicInSet` in output order.

reserved_sei_message_payload_byte is a byte reserved for future use by ITU-T | ISO/IEC.

D.3.26 No display SEI message semantics

The no display SEI message indicates that the current picture should not be displayed.

D.3.27 Time code SEI message semantics

The time code SEI message provides time code information similar to that defined by SMPTE ST 12-1 (2014) for field(s) or frame(s) of the current picture. When `vps_timing_info_present_flag` is equal to 1, it also defines a time offset for use in calculating a reference clock timestamp from the time code syntax elements to indicate a time of origin, capture, or alternative ideal display.

num_clock_ts specifies the number of sets of clock timestamp syntax elements that may be present for the current picture, the presence of each of which is specified by a syntax flag `clock_timestamp_flag[i]` for a corresponding value of *i*. When `frame_field_info_present_flag` is equal to 1, the *i*-th set of clock timestamp syntax elements applies to the *i*-th field or frame associated with the indicated display of the current picture in the order specified by `pic_struct` in Table D.2. The value of `num_clock_ts` shall not be equal to 0.

When `field_seq_flag` is equal to 1, the value of `num_clock_ts` shall be equal to 1.

When `frame_field_info_present_flag` is equal to 1, the value of `num_clock_ts` shall be equal to the value of `DeltaToDivisor` specified in Table E.7.

When `vps_timing_info_present_flag` is equal to 1, the contents of the clock timestamp syntax elements indicate a time of origin, capture, or alternative ideal display. This indicated time is computed as

$$\text{clockTimestamp}[i] = ((\text{hH} * 60 + \text{mM}) * 60 + \text{sS}) * \text{vui_time_scale} + \text{nFrames} * (\text{vui_num_units_in_tick} * (1 + \text{units_field_based_flag}[i])) + \text{tOffset} \quad (\text{D-26})$$

in units of clock ticks of a clock with clock frequency equal to `vui_time_scale` Hz, relative to some unspecified point in time for which `clockTimestamp[i]` would be equal to 0. Output order and DPB output timing are not affected by the value of `clockTimestamp[i]`. When `frame_field_info_present_flag` is equal to 1 and two or more pictures with `pic_struct` equal to 0 are consecutive in output order and have equal values of `clockTimestamp[i]`, the indication is that the pictures represent the same content and that the last such picture in output order is the preferred representation.

NOTE 1 – `clockTimestamp[i]` time indications may aid display on devices with refresh rates other than those well-matched to DPB output times. However, the time code SEI message does not affect the output order and DPB output timing defined in this Specification.

clock_timestamp_flag[i] equal to 1 indicates that associated set of clock timestamp syntax elements are present for the *i*-th set of clock timestamp syntax elements of the current picture. `clock_timestamp_flag[i]` equal to 0 indicates that the associated set of clock timestamp syntax elements is not present. When `vps_timing_info_present_flag` is equal to 1, `num_clock_ts` is greater than 1 and `clock_timestamp_flag[i]` is equal to 1 for more than one value of *i*, the value of `clockTimestamp[i]` shall be non-decreasing with increasing value of *i*.

For the *i*-th set of clock timestamp syntax elements of the current picture, the previous set of clock timestamp syntax elements in decoding order (or in output order, respectively), is defined as follows:

- If *i* is equal to 0, the previous set of clock timestamp syntax elements in decoding order (or in output order, respectively) is the set of syntax elements for the highest value of *i* for the previous picture in decoding order (or in output order, respectively) that contained a time code SEI message.
- Otherwise, the previous set of clock timestamp syntax elements in decoding order (or in output order, respectively) is the (*i* – 1)-th set of clock timestamp syntax elements of the current picture.

units_field_based_flag[i] is used in calculating `clockTimestamp[i]`, as specified in Equation D-26.

NOTE 2 – `units_field_based_flag[i]` should be the same for all values of *i* for all pictures in the CVS. When `field_seq_flag` is equal to 1 or `frame_field_info_present_flag` is equal to 1 and `pic_struct` is in the range of 1 to 6 or 9 to 12, inclusive, `units_field_based_flag[i]` should be equal to 1.

counting_type[i] specifies the method of dropping values of the `n_frames[i]` syntax element as specified in Table D.11.

NOTE 3 – `counting_type[i]` should be the same for all values of *i* for all pictures in the CVS.

Table D.11 – Definition of `counting_type[i]` values

Value	Interpretation
-------	----------------

0	No dropping of $n_frames[i]$ count values and no use of $time_offset_value[i]$
1	No dropping of $n_frames[i]$ count values
2	Dropping of individual zero values of $n_frames[i]$ count
3	Dropping of individual values of $n_frames[i]$ count equal to $MaxFPS - 1$
4	Dropping of the two lowest (value 0 and 1) $n_frames[i]$ counts when $seconds_value[i]$ is equal to 0 and $minutes_value[i]$ is not an integer multiple of 10
5	Dropping of unspecified individual $n_frames[i]$ count values
6	Dropping of unspecified numbers of unspecified $n_frames[i]$ count values
7..31	Reserved

full_timestamp_flag[i] equal to 1 specifies that the $n_frames[i]$ syntax element is followed by $seconds_value[i]$, $minutes_value[i]$ and $hours_value[i]$. **full_timestamp_flag[i]** equal to 0 specifies that the $n_frames[i]$ syntax element is followed by $seconds_flag[i]$.

discontinuity_flag[i] equal to 0 indicates that the difference between the current value of $clockTimestamp[i]$ and the value of $clockTimestamp[i]$ computed from the previous set of clock timestamp syntax elements in output order can be interpreted as the time difference between the times of origin or capture of the associated frames or fields. **discontinuity_flag[i]** equal to 1 indicates that the difference between the current value of $clockTimestamp[i]$ and the value of $clockTimestamp[i]$ computed from the previous set of clock timestamp syntax elements in output order should not be interpreted as the time difference between the times of origin or capture of the associated frames or fields. When **vps_timing_info_present_flag** is equal to 1 and **discontinuity_flag[i]** is equal to 0, the value of $clockTimestamp[i]$ shall be greater than or equal to the value of $clockTimestamp[i]$ for the previous set of clock timestamp syntax elements in output order (when present).

cnt_dropped_flag[i] specifies the skipping of one or more values of $n_frames[i]$ using the counting method specified by **counting_type[i]**.

n_frames[i] specifies the value of $nFrames$ used to compute $clockTimestamp[i]$. When **vps_timing_info_present_flag** is equal to 1, $n_frames[i]$ shall be less than $MaxFPS$, where $MaxFPS$ is specified by

$$MaxFPS = \text{Ceil}(vui_time_scale \div ((1 + units_field_based_flag[i]) * vui_num_units_in_tick)) \quad (D-27)$$

NOTE 4 – $n_frames[i]$ is a frame-based counter. To provide field-specific timing indications when the current picture is a frame, $time_offset_value[i]$ should be used to indicate a distinct $clockTimestamp[i]$ for each field of the frame.

When **counting_type[i]** is equal to 2 and **cnt_dropped_flag[i]** is equal to 1, $n_frames[i]$ shall be equal to 1 and the value of $n_frames[i]$ for the previous set of clock timestamp syntax elements in output order (when present) shall not be equal to 0 unless **discontinuity_flag[i]** is equal to 1.

NOTE 5 – When **counting_type[i]** is equal to 2, the need for increasingly large magnitudes of $tOffset$ in Equation D-26 when using fixed non-integer frame rates (e.g., 12.5 frames per second with vui_time_scale equal to 50 and $vui_num_units_in_tick$ equal to 2 and **units_field_based_flag[i]** equal to 0) can be avoided by occasionally skipping over the value $n_frames[i]$ equal to 0 when counting (e.g., counting $n_frames[i]$ from 0 to 12, then incrementing $seconds_value[i]$ and counting $n_frames[i]$ from 1 to 12, then incrementing $seconds_value[i]$ and counting $n_frames[i]$ from 0 to 12, etc.).

When **vps_timing_info_present_flag** is equal to 1, **counting_type[i]** is equal to 3 and **cnt_dropped_flag[i]** is equal to 1, $n_frames[i]$ shall be equal to 0 and the value of $n_frames[i]$ for the previous set of clock timestamp syntax elements in output order (when present) shall not be equal to $MaxFPS - 1$ unless **discontinuity_flag[i]** is equal to 1.

NOTE 6 – When **counting_type[i]** is equal to 3, the need for increasingly large magnitudes of $tOffset$ in Equation D-26 when using fixed non-integer frame rates (e.g., 12.5 frames per second with vui_time_scale equal to 50 and $vui_num_units_in_tick$ equal to 2 and **units_field_based_flag[i]** equal to 0) can be avoided by occasionally skipping over the value $n_frames[i]$ equal to $MaxFPS - 1$ when counting (e.g., counting $n_frames[i]$ from 0 to 12, then incrementing $seconds_value[i]$ and counting $n_frames[i]$ from 0 to 11, then incrementing $seconds_value[i]$ and counting $n_frames[i]$ from 0 to 12, etc.).

When **counting_type[i]** is equal to 4 and **cnt_dropped_flag[i]** is equal to 1, $n_frames[i]$ shall be equal to 2 and $seconds_value[i]$ shall be equal to zero and $minutes_value[i]$ shall not be an integer multiple of 10 and $n_frames[i]$ for the previous set of clock timestamp syntax elements in output order (when present) shall not be equal to 0 or 1 unless **discontinuity_flag[i]** is equal to 1.

NOTE 7 – When **counting_type[i]** is equal to 4, the need for increasingly large magnitudes of $tOffset$ in Equation D-26 when using fixed non-integer frame rates (e.g., $30000 \div 1001$ frames per second with vui_time_scale equal to 60 000 and $vui_num_units_in_tick$ equal to 1 001 and **units_field_based_flag[i]** equal to 1) can be reduced by occasionally skipping over the values of $n_frames[i]$

equal to 0 and 1 when counting (e.g., counting `n_frames[i]` from 0 to 29, then incrementing `seconds_value[i]` and counting `n_frames[i]` from 0 to 29, etc., until the `seconds_value[i]` is zero and `minutes_value[i]` is not an integer multiple of ten, then counting `n_frames[i]` from 2 to 29, then incrementing `seconds_value[i]` and counting `n_frames[i]` from 0 to 29, etc.). This counting method is well known in the industry and is often referred to as "NTSC drop-frame" counting.

When `vps_timing_info_present_flag` is equal to 1, `counting_type[i]` is equal to 5 or 6 and `cnt_dropped_flag[i]` is equal to 1, `n_frames[i]` shall not be equal to 1 plus the value of `n_frames[i]` for the previous set of clock timestamp syntax elements in output order (when present) modulo `MaxFPS` unless `discontinuity_flag[i]` is equal to 1.

NOTE 8 – When `counting_type[i]` is equal to 5 or 6, the need for increasingly large magnitudes of `tOffset` in Equation D-26 when using fixed non-integer frame rates can be avoided by occasionally skipping over some values of `n_frames[i]` when counting. The specific values of `n_frames[i]` that are skipped are not specified when `counting_type[i]` is equal to 5 or 6.

`seconds_flag[i]` equal to 1 specifies that `seconds_value[i]` and `minutes_flag[i]` are present when `full_timestamp_flag[i]` is equal to 0. `seconds_flag[i]` equal to 0 specifies that `seconds_value[i]` and `minutes_flag[i]` are not present.

`seconds_value[i]` specifies the value of `sS` used to compute `clockTimestamp[i]`. The value of `seconds_value[i]` shall be in the range of 0 to 59, inclusive. When `seconds_value[i]` is not present, its value is inferred to be equal to the value of `seconds_value[i]` for the previous set of clock timestamp syntax elements in decoding order, and it is required that such a previous `seconds_value[i]` shall have been present.

`minutes_flag[i]` equal to 1 specifies that `minutes_value[i]` and `hours_flag[i]` are present when `full_timestamp_flag[i]` is equal to 0 and `seconds_flag[i]` is equal to 1. `minutes_flag[i]` equal to 0 specifies that `minutes_value[i]` and `hours_flag[i]` are not present.

`minutes_value[i]` specifies the value of `mM` used to compute `clockTimestamp[i]`. The value of `minutes_value[i]` shall be in the range of 0 to 59, inclusive. When `minutes_value[i]` is not present, its value is inferred to be equal to the value of `minutes_value[i]` for the previous set of clock timestamp syntax elements in decoding order, and it is required that such a previous `minutes_value[i]` shall have been present.

`hours_flag[i]` equal to 1 specifies that `hours_value[i]` is present when `full_timestamp_flag[i]` is equal to 0 and `seconds_flag[i]` is equal to 1 and `minutes_flag[i]` is equal to 1.

`hours_value[i]` specifies the value of `hH` used to compute `clockTimestamp[i]`. The value of `hours_value[i]` shall be in the range of 0 to 23, inclusive. When `hours_value[i]` is not present, its value is inferred to be equal to the value of `hours_value[i]` for the previous set of clock timestamp syntax elements in decoding order, and it is required that such a previous `hours_value[i]` shall have been present.

`time_offset_length[i]` greater than 0 specifies the length in bits of the `time_offset_value[i]` syntax element. `time_offset_length[i]` equal to 0 specifies that the `time_offset_value[i]` syntax element is not present. When `counting_type[i]` is equal to 0, `time_offset_length[i]` shall be equal to 0.

NOTE 9 – `time_offset_length[i]` should be the same for all values of `i` for all pictures in the CVS.

`time_offset_value[i]` specifies the value of `tOffset` used to compute `clockTimestamp[i]`. The number of bits used to represent `time_offset_value[i]` is equal to `time_offset_length[i]`. When `time_offset_value[i]` is not present, its value is inferred to be equal to 0.

D.3.28 Mastering display colour volume SEI message semantics

This SEI message identifies the colour volume (the colour primaries, white point, and luminance range) of a display considered to be the mastering display for the associated video content – e.g., the colour volume of a display that was used for viewing while authoring the video content. The described mastering display is a three-colour additive display system that has been configured to use the indicated mastering colour volume.

This SEI message does not identify the measurement methodologies and procedures used for determining the indicated values or provide any description of the mastering environment. It also does not provide information on colour transformations that would be appropriate to preserve creative intent on displays with colour volumes different from that of the described mastering display.

The information conveyed in this SEI message is intended to be adequate for purposes corresponding to the use of SMPTE ST 2086 (2018).

When a mastering display colour volume SEI message is present for any picture of a CLVS of a particular layer, a mastering display colour volume SEI message shall be present for the first picture of the CLVS. The mastering display colour volume SEI message persists for the current layer in decoding order from the current picture until the end of the CLVS. All mastering display colour volume SEI messages that apply to the same CLVS shall have the same content.

`display_primaries_x[c]`, when in the range of 5 to 37 000, inclusive, specifies the normalized `x` chromaticity coordinate of the colour primary component `c` of the mastering display, according to the CIE 1931 definition of `x` as specified in ISO

11664-1 (see also ISO 11664-3 and CIE 15), in increments of 0.00002. When `display primaries_x[c]` is not in the range of 5 to 37 000, inclusive, the normalized x chromaticity coordinate of the colour primary component c of the mastering display is unknown or unspecified or specified by other means not specified in this Specification.

display_primaries_y[c], when in the range of 5 to 42 000, inclusive, specifies the normalized y chromaticity coordinate of the colour primary component c of the mastering display, according to the CIE 1931 definition of y as specified in ISO 11664-1 (see also ISO 11664-3 and CIE 15), in increments of 0.00002. When `display_primaries_y[c]` is not in the range of 5 to 42 000, inclusive, the normalized y chromaticity coordinate of the colour primary component c of the mastering display is unknown or unspecified or specified by other means not specified in this Specification.

For describing mastering displays that use red, green, and blue colour primaries, it is suggested that index value c equal to 0 should correspond to the green primary, c equal to 1 should correspond to the blue primary, and c equal to 2 should correspond to the red colour primary (see also Annex E and Table E.3).

white_point_x, when in the range of 5 to 37 000, inclusive, specifies the normalized x chromaticity coordinate of the white point of the mastering display, according to the CIE 1931 definition of x as specified in ISO 11664-1 (see also ISO 11664-3 and CIE 15), in normalized increments of 0.00002. When `white_point_x` is not in the range of 5 to 37 000, inclusive, the normalized x chromaticity coordinate of the white point of the mastering display is indicated to be unknown or unspecified or specified by other means not specified in this Specification.

white_point_y, when in the range of 5 to 42 000, inclusive, specifies the normalized y chromaticity coordinate of the white point of the mastering display, according to the CIE 1931 definition of y as specified in ISO 11664-1 (see also ISO 11664-3 and CIE 15), in normalized increments of 0.00002. When `white_point_y` is not in the range of 5 to 42 000, inclusive, the normalized y chromaticity coordinate of the white point of the mastering display is indicated to be unknown or unspecified or specified by other means not specified in this Specification.

NOTE 1 – SMPTE ST 2086 (2018) specifies that the normalized x and y chromaticity coordinate values for the mastering display colour primaries and white point are to be represented with four decimal places. This would correspond with using values of the syntax elements `display_primaries_x[c]`, `display_primaries_y[c]`, `white_point_x`, and `white_point_y`, as defined in this Specification, that are multiples of 5.

NOTE 2 – An example of the use of values outside the range for which semantics are specified in this Specification is that ANSI/CTA 861-G (2016) uses normalized (x, y) chromaticity coordinate values of (0,0) for the white point to indicate that the white point chromaticity is unknown.

max_display_mastering_luminance, when in the range of 50 000 to 100 000 000, specifies the nominal maximum display luminance of the mastering display in units of 0.0001 candelas per square metre. When `max_display_mastering_luminance` is not in the range of 50 000 to 100 000 000, the nominal maximum display luminance of the mastering display is indicated to be unknown or unspecified or specified by other means not specified in this Specification.

NOTE 3 – SMPTE ST 2086 (2018) specifies that the nominal maximum display luminance of the mastering display is to be specified as a multiple of 1 candela per square meter. This would correspond with using values of the syntax element `max_display_mastering_luminance`, as defined in this Specification, that are a multiple of 10 000.

NOTE 4 – An example of the use of values outside the range for which semantics are specified in this Specification is that ANSI/CTA 861-G (2016) uses the value 0 for the nominal maximum display luminance of the mastering display to indicate that the nominal maximum display luminance of the mastering display is unknown.

min_display_mastering_luminance, when in the range of 1 to 50 000, specifies the nominal minimum display luminance of the mastering display in units of 0.0001 candelas per square metre. When `min_display_mastering_luminance` is not in the range of 1 to 50 000, the nominal maximum display luminance of the mastering display is unknown or unspecified or specified by other means not specified in this Specification. When `max_display_mastering_luminance` is equal to 50 000, `min_display_mastering_luminance` shall not be equal to 50 000.

NOTE 5 – SMPTE ST 2086 (2018) specifies that the nominal minimum display luminance of the mastering display is to be specified as a multiple of 0.0001 candelas per square metre, which corresponds to the semantics specified in this Specification.

NOTE 6 – An example of the use of values outside the range for which semantics are specified in this Specification is that ANSI/CTA 861-G (2016) uses the value 0 for the nominal minimum display luminance of the mastering display to indicate that the nominal minimum display luminance of the mastering display is unknown.

NOTE 7 – Another example of the potential use of values outside the range for which semantics are specified in this Specification is that SMPTE ST 2086 (2018) indicates that values outside the specified range could be used to indicate that the black level and contrast of the mastering display have been adjusted using picture line-up generation equipment (PLUGE).

At the minimum luminance, the mastering display is considered to have the same nominal chromaticity as the white point.

D.3.29 Segmented rectangular frame packing arrangement SEI message semantics

This SEI message informs the decoder that the output cropped decoded picture contains samples of multiple distinct spatially packed constituent frames that are packed into one frame using a rectangular region frame packing arrangement. This information can be used by the decoder to appropriately rearrange the samples and process the samples of the constituent frames appropriately for display or other purposes (which are outside the scope of this Specification).

Each colour component plane of the output cropped decoded pictures contains a rectangular region frame packing arrangement of corresponding planes of two constituent frames as illustrated in Figure D.10.

NOTE 1 – Figure D.10 provides an illustration of the rearrangement process for the rectangular region frame packing arrangement.

This SEI message may be associated with pictures that are either frames (when `field_seq_flag` is equal to 0) or fields (when `field_seq_flag` is equal to 1). The rectangular region frame packing arrangement of the samples is specified in terms of the sampling structure of a frame in order to define a frame packing arrangement structure that is invariant with respect to whether a picture is a single field of such a packed frame or is a complete packed frame.

When `general_non_packed_constraint_flag` is equal to 1 in the active SPS for the current layer, there shall be no segmented rectangular frame packing arrangement SEI messages applicable for any picture of the CLVS of the current layer.

When a frame packing arrangement SEI message is applicable for any picture of the CLVS of the current layer, there shall be no segmented rectangular frame packing arrangement SEI messages applicable for any picture of the CLVS of the current layer.

`segmented_rect_frame_packing_arrangement_cancel_flag` equal to 1 indicates that the SEI message cancels the persistence of any previous segmented rectangular frame packing arrangement SEI message in output order. `segmented_rect_frame_packing_arrangement_cancel_flag` equal to 0 indicates that rectangular region frame packing arrangement information follows.

`segmented_rect_content_interpretation_type` indicates the intended interpretation of the constituent frames as specified in Table D.12. Values of `segmented_rect_content_interpretation_type` that do not appear in Table D.12 are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore rectangular region frame packing arrangement SEI messages that contain reserved values of `segmented_rect_content_interpretation_type`.

For the specified frame packing arrangement scheme, there are two constituent frames that are referred to as frame 0 and frame 1.

Table D.12 – Definition of `segmented_rect_content_interpretation_type`

Value	Interpretation
0	Unspecified relationship between the frame packed constituent frames
1	Indicates that the two constituent frames form the left and right views of a stereo view scene, with frame 0 being associated with the left view and frame 1 being associated with the right view
2	Indicates that the two constituent frames form the right and left views of a stereo view scene, with frame 0 being associated with the right view and frame 1 being associated with the left view

NOTE 2 – The value 2 for `segmented_rect_content_interpretation_type` is not expected to be prevalent in industry use of this SEI message. However, the value was specified herein for purposes of completeness.

`segmented_rect_frame_packing_arrangement_persistence_flag` specifies the persistence of the segmented rectangular frame packing arrangement SEI message for the current layer.

`segmented_rect_frame_packing_arrangement_persistence_flag` equal to 0 specifies that the rectangular region frame packing arrangement SEI message applies to the current decoded frame only.

Let `picA` be the current picture. `segmented_rect_frame_packing_arrangement_persistence_flag` equal to 1 specifies that the segmented rectangular frame packing arrangement SEI message persists for the current layer in output order until any of the following conditions are true:

- A new CLVS of the current layer begins.
- The bitstream ends.
- A frame `picB` in the current layer in an access unit containing a segmented rectangular frame packing arrangement SEI message applicable to the current layer is output for which `PicOrderCnt(picB)` is greater than `PicOrderCnt(picA)`, where `PicOrderCnt(picB)` and `PicOrderCnt(picA)` are the `PicOrderCntVal` values of `picB` and `picA`, respectively, immediately after the invocation of the decoding process for picture order count for `picB`.

NOTE 3 – The default display window parameters in the VUI parameters of the SPS should be used by an encoder to indicate to a decoder that does not interpret the rectangular region frame packing arrangement SEI message that the default display window is an area within only one of the two constituent frames.

Let `croppedWidth` and `croppedHeight` be the width and height, respectively, of the cropped frame output from the decoder in units of luma samples, derived as follows:

$$\text{croppedWidth} = \text{pic_width_in_luma_samples} - \text{SubWidthC} * (\text{conf_win_right_offset} + \text{conf_win_left_offset}) \quad (\text{D-28})$$

$$\text{croppedHeight} = \text{pic_height_in_luma_samples} - \text{SubHeightC} * (\text{conf_win_bottom_offset} + \text{conf_win_top_offset}) \quad (\text{D-29})$$

It is a requirement of bitstream conformance for the rectangular region frame packing arrangement that `croppedWidth` and `croppedHeight` shall be integer multiples of 3.

Let `oneThirdWidth` and `oneThirdHeight` be derived as follows:

$$\text{oneThirdWidth} = \text{croppedWidth} / 3 \quad (\text{D-30})$$

$$\text{oneThirdHeight} = \text{croppedHeight} / 3 \quad (\text{D-31})$$

The rectangular region frame packing arrangement is composed of five rectangular regions identified as R0, R1, R2, R3 and R4 as illustrated in Figure D.10.

The width and height of the region R0 are specified in units of frame luma samples as follows:

$$\text{r0}_W = 2 * \text{oneThirdWidth} \quad (\text{D-32})$$

$$\text{r0}_H = 2 * \text{oneThirdHeight} \quad (\text{D-33})$$

The width and height of the region R1 are specified in units of frame luma samples as follows:

$$\text{r1}_W = \text{oneThirdWidth} \quad (\text{D-34})$$

$$\text{r1}_H = 2 * \text{oneThirdHeight} \quad (\text{D-35})$$

The width and height of the region R2 are specified in units of frame luma samples as follows:

$$\text{r2}_W = \text{oneThirdWidth} \quad (\text{D-36})$$

$$\text{r2}_H = \text{oneThirdHeight} \quad (\text{D-37})$$

The width and height of the region R3 are specified in units of frame luma samples as follows:

$$\text{r3}_W = \text{oneThirdWidth} \quad (\text{D-38})$$

$$\text{r3}_H = \text{oneThirdHeight} \quad (\text{D-39})$$

The width and height of the region R4 are specified in units of frame luma samples as follows:

$$\text{r4}_W = \text{oneThirdWidth} \quad (\text{D-40})$$

$$\text{r4}_H = \text{oneThirdHeight} \quad (\text{D-41})$$

Constituent frame 0 is obtained by cropping from the decoded frames the region R0 and constituent frame 1 is obtained by stacking vertically the regions R2 and R3 and placing the resulting rectangle to the right of the region R1. The region R4 is not part of either constituent frame and is discarded.

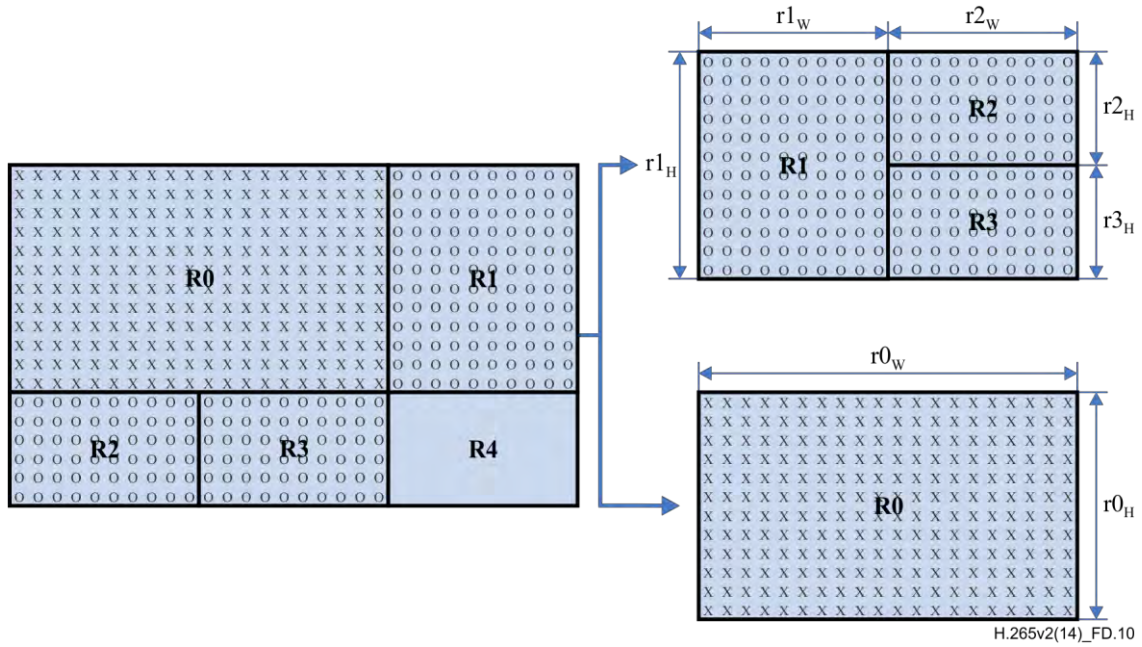


Figure D.10 – Rearrangement of a segmented rectangular frame packing arrangement

D.3.30 Temporal motion-constrained tile sets SEI message semantics

The temporal motion-constrained tile sets SEI message indicates that the following constraints apply:

- No sample values outside each identified tile set or outside the picture are referenced for inter prediction.
- For PUs located directly left of the right tile boundary of each identified tile set except the last one at the bottom right, the following applies when $\text{CuPredMode}[xPb][yPb]$ is equal to MODE_INTER , where (xPb, yPb) specifies the top-left sample of the corresponding luma prediction block relative to the top-left sample of the current picture:
 - With the number of spatial merging candidates $\text{numSpatialMergeCand}$ derived as follows:

$$\begin{aligned} \text{numSpatialMergeCand} = & \text{availableFlagA}_0 + \text{availableFlagA}_1 + \\ & \text{availableFlagB}_0 + \text{availableFlagB}_1 + \text{availableFlagB}_2 \end{aligned} \quad (\text{D-42})$$

where availableFlagA_0 , availableFlagA_1 , availableFlagB_0 , availableFlagB_1 , and availableFlagB_2 are the output of the derivation process for spatial merging candidates specified in clause 8.5.3.2.3, the following applies:

- If $\text{numSpatialMergeCand}$ is equal to 0, $\text{merge_flag}[xPb][yPb]$ is equal to 0.
- Otherwise ($\text{numSpatialMergeCand}$ is greater than 0), $\text{merge_idx}[xPb][yPb]$ is in the range of 0 to $\text{numSpatialMergeCand} - 1$, inclusive.
- With the number of spatial motion vector predictor candidates numSpatialMvpCand derived as follows:

$$\begin{aligned} \text{if}(\text{availableFlagLXA}) \\ \text{numSpatialMvpCand} = & \text{availableFlagLXA} + ((\text{mvLXA} \neq \text{mvLXB}) ? \text{availableFlagLXB} : 0) \\ \text{else} \\ \text{numSpatialMvpCand} = & \text{availableFlagLXB} \end{aligned} \quad (\text{D-43})$$

where availableFlagLXA , availableFlagLXB , mvLXA , and mvLXB are the output of the derivation process for motion vector predictor candidates from neighbouring prediction unit partitions specified in clause 8.5.3.2.7, the following applies:

- If numSpatialMvpCand is equal to 0, $\text{mvp_l0_flag}[xPb][yPb]$ and $\text{mvp_l1_flag}[xPb][yPb]$ are equal to 1.
- Otherwise (numSpatialMvpCand is greater than 0), $\text{mvp_l0_flag}[xPb][yPb]$ and $\text{mvp_l1_flag}[xPb][yPb]$ are in the range of 0 to $\text{numSpatialMvpCand} - 1$, inclusive.

NOTE 1 – The first constraint restricts motion vector values to only those that refer either to full-sample locations inside each identified tile set or to fractional-sample locations that require only full-sample locations inside each identified tile set for

interpolation. The second constraint prohibits the usage of motion vector candidates for temporal motion vector prediction that are derived from blocks outside each identified tile set.

Let a set of pictures associatedPicSet be the pictures with nuh_layer_id equal to targetLayerId from the access unit containing the SEI message, inclusive, up to but not including the first of any of the following in decoding order:

- The next access unit, in decoding order, that contains a temporal motion-constrained tile sets SEI message applicable to targetLayerId.
- The next IRAP picture with NoRaslOutputFlag equal to 1, in decoding order, with nuh_layer_id equal to targetLayerId.
- The next IRAP access unit, in decoding order, with NoClasOutputFlag equal to 1.

The scope of the temporal motion-constrained tile sets SEI message is the set of pictures associatedPicSet.

When a temporal motion-constrained tile sets SEI message is present for any picture in associatedPicSet, a temporal motion-constrained tile sets SEI message applicable to targetLayerId shall be present for the first picture of associatedPicSet in decoding order and may also be present for other pictures of associatedPicSet.

When a temporal motion-constrained tile sets SEI message applicable to targetLayerId is present for a picture in associatedPicSet and tiles_enabled_flag is equal to 0 for the PPS that is active for the picture, the picture contains only one tile, which forms the only MCTS, and the values of mc_all_tiles_exact_sample_value_match_flag and each_tile_one_tile_set_flag shall both be equal to 1.

The temporal motion-constrained tile sets SEI message applicable to targetLayerId shall not be present for any picture in associatedPicSet unless every PPS that is active for any picture in associatedPicSet has the same values of the syntax elements num_tile_columns_minus1, num_tile_rows_minus1, uniform_spacing_flag, column_width_minus1[i] and row_height_minus1[i].

NOTE 2 – This constraint is similar to the constraint associated with tiles_fixed_structure_flag equal to 1 and it may be desirable for tiles_fixed_structure_flag to be equal to 1 when the temporal motion-constrained tile sets SEI message is present (although this is not required).

NOTE 3 – When loop filtering is applied across tile boundaries, inter prediction of any samples within a temporal motion-constrained tile set that refers to samples within four samples from a temporal motion-constrained tile set boundary that is not also a picture boundary may result in propagation of mismatch error. An encoder can avoid such potential error propagation by avoiding the use of motion vectors that cause such references.

When more than one temporal motion-constrained tile sets SEI message applicable to targetLayerId is present for the pictures of associatedPicSet, they shall contain identical content.

When a temporal motion-constrained tile sets SEI message is present, a slice segment that contains one or more tiles in any particular temporal motion-constrained tile set shall not be a dependent slice segment of an independent slice segment that contains one or more tiles that do not belong to that temporal motion-constrained tile set.

For purposes of referencing a particular temporal motion-constrained tile set that is identified in a temporal motion-constrained tile sets SEI message (e.g., for use with a motion-constrained tile sets extraction information sets SEI message or a motion-constrained tile sets extraction information nesting SEI message), an MCTS index is defined as follows:

- If the value of each_tile_one_tile_set_flag of the temporal motion-constrained tile sets SEI message is equal to 0, the MCTS index is the value of the variable i within the loop of the num_sets_in_message_minus1 + 1 sets of MCTS information specified by the temporal MCTS SEI message.
- Otherwise, the MCTS index of each MCTS is the tile position of the tile in tile raster scan order.

mc_all_tiles_exact_sample_value_match_flag equal to 0 indicates that when the CTUs that do not belong to any tile in the motion-constrained tile sets are not decoded and the boundaries of the tiles in the motion-constrained tile sets are treated as picture boundaries for purposes of the decoding process, the decoded value of each sample in the tiles in the motion-constrained tile sets may not be exactly the same as the decoded value of the same sample when all the CTUs of the picture are decoded. mc_all_tiles_exact_sample_value_match_flag equal to 1 indicates that when the CTUs that do not belong to any tile in the motion-constrained tile sets are not decoded and the boundaries of the tiles in the motion-constrained tile sets are treated as picture boundaries for purposes of the decoding process, the decoded value of each sample in the tiles in the motion-constrained tile sets is exactly the same as the decoded value of the sample that would be obtained when all the CTUs of all pictures in associatedPicSet are decoded.

each_tile_one_tile_set_flag equal to 1 indicates that each and every tile in the picture is included in a separate temporal motion-constrained tile set. each_tile_one_tile_set_flag equal to 0 indicates that such constraint may not be applied.

limited_tile_set_display_flag equal to 1 specifies that the display_tile_set_flag[i] syntax element is present and indicates that the tiles not included within any tile set with display_tile_set_flag[i] equal to 1 are not intended for display. limited_tile_set_display_flag equal to 0 specifies that the display_tile_set_flag[i] syntax element is not present.

num_sets_in_message_minus1 plus 1 specifies the number of temporal motion-constrained tile sets identified in the SEI message. The value of **num_sets_in_message_minus1** shall be in the range of 0 to 255, inclusive.

mcts_id[i] contains an identifying number that may be used to identify the purpose of the *i*-th identified tile set (for example, to identify an area to be extracted from associatedPicSet for a particular purpose). The value of **mcts_id[i]** shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of **mcts_id[i]** from 0 to 255, inclusive, and from 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of **mcts_id[i]** from 256 to 511, inclusive, and from 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC and bitstreams shall not contain such values. Decoders encountering a value of **mcts_id[i]** in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

display_tile_set_flag[i] equal to 1 indicates that the *i*-th tile set is intended for display. **display_tile_set_flag[i]** equal to 0 indicates that the *i*-th tile set is not intended for display. When not present, the value of **display_tile_set_flag[i]** is inferred to be equal to 1.

num_tile_rects_in_set_minus1[i] plus 1 specifies the number of rectangular regions of tiles in the *i*-th identified temporal motion-constrained tile set. The value of **num_tile_rects_in_set_minus1[i]** shall be in the range of 0 to $(\text{num_tile_columns_minus1} + 1) * (\text{num_tile_rows_minus1} + 1) - 1$, inclusive.

top_left_tile_idx[i][j] and **bottom_right_tile_idx[i][j]** identify the tile position of the top-left tile and the tile position of the bottom-right tile in a rectangular region of the *i*-th identified temporal motion-constrained tile set, respectively, in tile raster scan order.

The value of **top_left_tile_idx[i][j]** and **bottom_right_tile_idx[i][j]** shall be in the range of 0 to $(\text{num_tile_columns_minus1} + 1) * (\text{num_tile_rows_minus1} + 1) - 1$, inclusive.

mc_exact_sample_value_match_flag[i] equal to 0 indicates that when the CTUs that are outside of the *i*-th identified temporal motion-constrained tile set are not decoded and the boundaries of the temporal motion-constrained tile set are treated as picture boundaries for purposes of the decoding process, the value of each sample in the identified tile set may not be exactly the same as the value of the same sample when all the CTUs of the picture are decoded. **mc_exact_sample_value_match_flag[i]** equal to 1 indicates that when the CTUs that do not belong to the temporal motion-constrained tile set are not decoded and the boundaries of the temporal motion-constrained tile set are treated as picture boundaries for purposes of the decoding process, the value of each sample in the temporal motion-constrained tile set would be exactly the same as the value of the sample that would be obtained when all the CTUs of all pictures in associatedPicSet are decoded.

NOTE 4 – It should be feasible to use **mc_exact_sample_value_match_flag[i]** equal to 1 when using certain combinations of **loop_filter_across_tiles_enabled_flag**, **pps_loop_filter_across_slices_enabled_flag**, **pps_deblocking_filter_disabled_flag**, **slice_loop_filter_across_slices_enabled_flag**, **slice_deblocking_filter_disabled_flag**, **sample_adaptive_offset_enabled_flag**, **slice_sao_luma_flag** and **slice_sao_chroma_flag**.

mcts_tier_level_idc_present_flag[i] equal to 1 specifies that the **mcts_tier_flag[i]** and **mcts_level_idc[i]** syntax elements are present. **mcts_tier_level_idc_present_flag[i]** equal to 0 specifies that the **mcts_tier_flag[i]** and **mcts_level_idc[i]** syntax elements are not present. When **mcts_tier_level_idc_present_flag[i]** is equal to 1, **num_tile_rects_in_set_minus1[i]** shall be equal to 0 and a slice segment containing one or more CTUs within the *i*-th motion-constrained tile set shall not include any CTU outside the *i*-th motion-constrained tile set. When not present, **mcts_tier_level_idc_present_flag[i]** is inferred to be equal to 0.

mcts_tier_flag[i] specifies the tier context for the interpretation of **mcts_level_idc[i]** corresponding to the *i*-th motion-constrained tile set. **mcts_tier_flag[i]** equal to 0 indicates conformance to the Main tier, and **mcts_tier_flag[i]** equal to 1 indicates conformance to the High tier, according to the tier constraints specified in Table A.8. **mcts_tier_flag[i]** shall be equal to 0 when **mcts_level_idc[i]** is less than 120. When not present, the value of **mcts_tier_flag[i]** is inferred to be equal to **general_tier_flag**.

mcts_level_idc[i] indicates a level to which the *i*-th tile set region, corresponding to the *i*-th motion-constrained tile set, conforms. The value of **mcts_level_idc[i]** shall be less than or equal to the value of **general_level_idc** in the active SPS RBSP. When not present, the value of **mcts_level_idc[i]** is inferred to be equal to **general_level_idc**.

mcts_max_tier_level_idc_present_flag equal to 1 specifies that the **mcts_max_tier_flag** and **mcts_max_level_idc** syntax elements are present. **mcts_max_tier_level_idc_present_flag** equal to 0 specifies that the **mcts_max_tier_flag** and **mcts_max_level_idc** syntax elements are not present. When **mcts_max_tier_level_idc_present_flag** is equal to 1, a slice segment containing one or more CTUs within a tile shall not include any CTU outside the tile. When not present, **mcts_max_tier_level_idc_present_flag** is inferred to be equal to 0.

mcts_max_tier_flag specifies the tier context for the interpretation of **mcts_max_level_idc** to which all motion-constrained tile sets conform. **mcts_max_tier_flag** equal to 0 indicates conformance to the Main tier, and **mcts_max_tier_flag** equal to 1 indicates conformance to the High tier, according to the tier constraints specified in Table A.8. **mcts_max_tier_flag** shall be equal to 0 when **mcts_max_level_idc** is less than 120. When not present, the value of **mcts_max_tier_flag** is inferred to be equal to **general_tier_flag**.

mcts_max_level_idc indicates a level to which all motion-constrained tile sets conform. The value of **mcts_max_level_idc** shall be less than or equal to the value of **general_level_idc** in the active SPS RBSP. When not present, the value of **mcts_max_level_idc** is inferred to be equal to **general_level_idc**.

The following describes the tier and level restrictions on the bitstreams of each motion-constrained tile set:

- If **mcts_tier_level_idc_present_flag[i]** and **mcts_max_tier_level_idc_present_flag** are both equal to 0, the **mctsLevelIdc[i]** of all motion-constrained tile sets are inferred to be equal to **general_level_idc** and the **mctsTierFlag[i]** of all motion-constrained tile sets are inferred to be equal to **general_tier_flag** and the specifications of Annex A apply to all motion-constrained tile sets.
- Otherwise (**mcts_tier_level_idc_present_flag[i]** or **mcts_max_tier_level_idc_present_flag** is equal to 1), the following applies:
 - The variables **mctsLevelIdc[i]**, **mctsTierFlag[i]**, **mctsWidthInSamplesY[i]**, **mctsHeightInSamplesY[i]**, **NumTileColumnsInMCTS[i]** and **NumTileRowsInMCTS[i]** are derived as follows:
 - If **each_tile_one_tile_set_flag** is equal to 0, for each tile set with index *i* in the range of 0 to **num_sets_in_message_minus1**, inclusive, the following applies:
 - **mctsLevelIdc[i]** is set equal to **mcts_level_idc[i]**.
 - **mctsTierFlag[i]** is set equal to **mcts_tier_flag[i]**.
 - **mctsWidthInSamplesY[i]** is set equal to the sum of **ColumnWidthInLumaSamples[k]**, for all *k* in the range of (**top_left_tile_idx[i][0] % (num_tile_rows_minus1 + 1)**) to (**bottom_right_tile_idx[i][0] % (num_tile_rows_minus1 + 1)**), inclusive.
 - **mctsHeightInSamplesY[i]** is set equal to the sum of **RowHeightInLumaSamples[k]**, for all *k* in the range of (**top_left_tile_idx[i][0] / (num_tile_rows_minus1 + 1)**) to (**bottom_right_tile_idx[i][0] / (num_tile_rows_minus1 + 1)**), inclusive.
 - **NumTileColumnsInMCTS[i]** is set equal to $(\text{bottom_right_tile_idx}[i][0] \% (\text{num_tile_rows_minus1} + 1)) - (\text{top_left_tile_idx}[i][0] \% (\text{num_tile_rows_minus1} + 1)) + 1$.
 - **NumTileRowsInMCTS[i]** is set equal to $(\text{bottom_right_tile_idx}[i][0] / (\text{num_tile_rows_minus1} + 1)) - (\text{top_left_tile_idx}[i][0] / (\text{num_tile_rows_minus1} + 1)) + 1$.
 - Otherwise (**each_tile_one_tile_set_flag** is equal to 1), for each tile with **TileId i**, with *i* in the range of 0 to $(\text{num_tile_columns_minus1} + 1) * (\text{num_tile_rows_minus1} + 1) - 1$, inclusive, the following applies:
 - **mctsLevelIdc[i]** is set equal to **mcts_max_level_idc**.
 - **mctsTierFlag[i]** is set equal to **mcts_max_tier_flag**.
 - **mctsWidthInSamplesY[i]** is set equal to **ColumnWidthInLumaSamples[i % (num_tile_rows_minus1 + 1)]**.
 - **mctsHeightInSamplesY[i]** is set equal to **RowHeightInLumaSamples[i / (num_tile_rows_minus1 + 1)]**.
 - **NumTileColumnsInMCTS[i]** is set equal to 1.
 - **NumTileRowsInMCTS[i]** is set equal to 1.
 - The variables **mctsSizeInSamplesY[i]** and **NumSliceSegmentsInMCTS[i]** are derived as follows:
 - **mctsSizeInSamplesY[i]** is set equal to **mctsWidthInSamplesY[i] * mctsHeightInSamplesY[i]**.
 - **NumSliceSegmentsInMCTS[i]** is set equal to the number of slice segments in the *i*-th motion-constrained tile set.
 - The variables **mctsMaxLumaPs[i]**, **mctsMaxCPB[i]**, **mctsMaxSliceSegments[i]**, **mctsMaxTileRows[i]**, **mctsMaxTileCols[i]**, **mctsMaxBR[i]** and **mctsMinCr[i]** are set equal to **MaxLumaPs**, **MaxCPB**, **MaxSliceSegmentsPerPicture**, **MaxTileRows**, **MaxTileCols** and **MaxBR**, respectively, specified in Table A.8 for the level indicated by **mctsLevelIdc[i]** with the tier indicated by **mctsTierFlag[i]**.
 - The variable **mctsMinCr[i]** is set equal to **MinCr** specified in Table A.9 for the level indicated by **mctsLevelIdc[i]** with the tier indicated by **mctsTierFlag[i]**.
 - **mctsLevelIdc[i]** and **mctsTierFlag[i]** indicate the level and tier to which the *i*-th motion-constrained tile set conforms, as specified in Annex A with the following modifications and additions:
 - The variable **PicSizeInSamplesY** is replaced with the variable **mctsSizeInSamplesY[i]**.
 - The value of **num_tile_columns_minus1** is replaced with the variable **NumTileColumnsInMCTS[i] - 1**.
 - The value of **num_tile_rows_minus1** is replaced with the variable **NumTileRowsInMCTS[i] - 1**.

- The number of slice segments for the i-th motion-constrained tile set is set to NumSliceSegmentsInMCTS[i].
- The variable MaxTileCols, MaxTileRows, MaxSliceSegmentsPerPicture, MaxLumaPs, MaxBR and MinCr are replaced with the variables mctsMaxTileCols[i], mctsMaxTileRows[i], mctsMaxSliceSegments[i], mctsMaxLumaPs[i], mctsMaxBR[i] and mctsMinCr[i], respectively.
- The nominal removal time of motion-constrained tile set from the CPB shall be the same as removal time of corresponding access unit.
- The difference between consecutive output times of motion-constrained tile sets from the DPB shall be same as difference between the output times of corresponding access unit.
- The value of NumBytesInNalUnit is replaced with the sum of the sizes of all NAL units belonging to the i-th motion-constrained tile set in bytes.

D.3.31 Chroma resampling filter hint SEI message semantics

The chroma resampling filter hint SEI message signals one downsampling process and one upsampling process for the chroma components of decoded pictures. When the sampling processes signalled in the chroma resampling filter hint SEI message are used, for any number of upsampling and downsampling iterations performed on the decoded pictures, the degradation of the colour components is expected to be minimized.

The chroma resampling filter hint SEI message shall not be present in a CLVS that has chroma_format_idc equal to 0.

It is a requirement of bitstream conformance that when a chroma resampling filter hint SEI message is present in a CLVS, chroma_sample_loc_type_top_field shall be equal to chroma_sample_loc_type_bottom_field in the same CLVS.

All chroma resampling filter hint SEI messages that apply to the same CLVS shall have the same content.

ver_chroma_filter_idc identifies the vertical components of the downsampling and upsampling sets of filters as specified in Table D.13. Based on the value of ver_chroma_filter_idc, the values of verFilterCoeff[][] are derived from Table D.18. The value of ver_chroma_filter_idc shall be in the range of 0 to 2, inclusive. Values of ver_chroma_filter_idc greater than 2 are reserved for future use by ITU-T | ISO/IEC.

When ver_chroma_filter_idc is equal to 0, the chroma resampling filter in the vertical direction is unspecified.

When chroma_format_idc is equal to 1, ver_chroma_filter_idc shall be equal to 1 or 2.

Table D.13 – ver_chroma_filter_idc values

Value	Description
0	Unspecified
1	Filters signalled by ver_filter_coeff[][]
2	Filters as described in SMPTE RP 2050-1 (2012)
>2	Reserved

hor_chroma_filter_idc identifies the horizontal components of the downsampling and upsampling sets of filters as specified in Table D.14. Based on the value of hor_chroma_filter_idc, the values of horFilterCoeff[][] are derived from Table D.19. The value of hor_chroma_filter_idc shall be in the range of 0 to 2, inclusive. Values of hor_chroma_filter_idc greater than 2 are reserved for future use by ITU-T | ISO/IEC.

When hor_chroma_filter_idc is equal to 0, the chroma resampling filter in the horizontal direction is unspecified.

When chroma_format_idc is equal to 3, hor_chroma_filter_idc shall be equal to 1 or 2.

When chroma_format_idc is equal to 2 and ver_chroma_filter_idc is equal to 2, hor_chroma_filter_idc shall be equal to 0.

It is a requirement of bitstream conformance that ver_chroma_filter_idc and hor_chroma_filter_idc shall not be both equal to 0.

Table D.14 – hor_chroma_filter_idc values

Value	Description
0	Unspecified
1	Filters signalled by hor_filter_coeff[][]
2	Filters as described in the 5/3 filter description of Rec. ITU-T T.800 ISO/IEC 15444-1
>2	Reserved

ver_filtering_field_processing_flag indicates whether the vertical operations of the downsampling and the upsampling sets of filters should be applied on a field-basis or a frame-basis. When **field_seq_flag** is equal to 1 and **pic_struct** is not within the range of 9 to 12 inclusive, **ver_filtering_field_processing_flag** shall be equal to 1.

Based on the values of **ver_filtering_field_processing_flag** and **field_seq_flag**, the following applies:

- If **ver_filtering_field_processing_flag** is equal to 1:
 - If **field_seq_flag** is equal to 1, each output field should be filtered independently.
 - Otherwise (**field_seq_flag** is equal to 0), for each decoded frame the filtering process should be applied to chroma samples belonging to the top field and to chroma samples belonging to the bottom field separately.
- Otherwise (**ver_filtering_field_processing_flag** is equal to 0):
 - If **field_seq_flag** is equal to 1, the filtering process should be performed on a frame basis. The input of the filtering process should be frames resulting from the interleaving of fields in accordance with the information conveyed by the **pic_struct** values.
 - Otherwise (**field_seq_flag** is equal to 0), the filtering process should be performed on a frame basis.

The variable **chromaSampleLocType** is derived as follows:

- If **chroma_format_idc** is equal to 1, **chromaSampleLocType** is set equal to **chroma_sample_loc_type_top_field**.
- Otherwise (**chroma_format_idc** is not equal to 1), **chromaSampleLocType** is set equal to 0.

When **chromaSampleLocType** is greater than 1, **ver_chroma_filter_idc** shall not be equal to 2.

When **chromaSampleLocType** is equal to 1, 3 or 5, **hor_chroma_filter_idc** shall not be equal to 2.

target_format_idc indicates the output sampling format of the chroma components (i.e., the position of chroma samples relative to that of the luma samples) after filtering, as specified in Table D.15. The value of **target_format_idc** shall be in the range of 1 to 3, inclusive. The value of **target_format_idc** shall not be equal to the value of **chroma_format_idc**.

When not present, the value of **target_format_idc** is inferred as follows:

- If **chroma_format_idc** is equal to 1 and **ver_chroma_filter_idc** is equal to 2, the following applies:
 - If **hor_chroma_filter_idc** is not equal to 2, the value of **target_format_idc** is inferred to be equal to 2.
 - Otherwise (**hor_chroma_filter_idc** is equal to 2), the value of **target_format_idc** is inferred to be equal to 3.
- Otherwise, if **chroma_format_idc** is equal to 2, the following applies:
 - If **ver_chroma_filter_idc** is equal to 2, the value of **target_format_idc** is inferred to be equal to 1.
 - Otherwise (**ver_chroma_filter_idc** is not equal to 2), the value of **target_format_idc** is inferred to be equal to 3.
- Otherwise, if **chroma_format_idc** is equal to 3, the following applies:
 - If **ver_chroma_filter_idc** is equal to 2, the value of **target_format_idc** is inferred to be equal to 1.
 - Otherwise (**ver_chroma_filter_idc** is not equal to 2), the value of **target_format_idc** is inferred to be equal to 2.

When **chroma_format_idc** is greater than 1 and **target_format_idc** is greater than 1, **ver_chroma_filter_idc** shall be equal to 0.

When **chroma_format_idc** is less than 3 and **target_format_idc** is less than 3, **hor_chroma_filter_idc** shall be equal to 0.

Table D.15 – Chroma sampling format indicated by target_format_idc

target_format_idc	Chroma sampling format
1	4:2:0
2	4:2:2
3	4:4:4

NOTE 1 – The logic associating sampling formats to numeric values of `target_format_idc` here is the same as the one used to associate sampling formats to numeric values of `chroma_format_idc`, as described in clause 6.2.

The variable `upsamplingFlag` is derived as follows:

- If `chroma_format_idc` is greater than `target_format_idc`, `upsamplingFlag` is set equal to 0.
- Otherwise (`chroma_format_idc` is less than `target_format_idc`), `upsamplingFlag` is set equal to 1.

num_vertical_filters specifies the number of filters signalled for chroma downsampling and upsampling in the vertical direction. When `ver_chroma_filter_idc` is equal to 1, depending on the values of `chromaSampleLocType` and `ver_filtering_field_processing_flag`, the value of `num_vertical_filters` shall be equal to the values specified in Table D.16.

Table D.16 – Constraints on the value of num_vertical_filters

Conditions		Mandatory value of num_vertical_filters
chromaSampleLocType	ver_filtering_field_processing_flag	
0, 1	0	2
	1	3
2, 3	0	3
	1	5
4, 5	0	3
	1	5

ver_tap_length_minus1[i] plus 1 specifies the length of the *i*-th filter in the vertical direction. The value of `ver_tap_length_minus1[i]` shall be in the range of 0 to 31, inclusive.

ver_filter_coeff[i][j] specifies the value of the *j*-th coefficient of the *i*-th filter in the vertical direction. The value of `ver_filter_coeff[i][j]` shall be in the range of $-2^{31} + 1$ to $2^{31} - 1$, inclusive.

The variable `verTapLength[]` is derived as follows:

- If `ver_chroma_filter_idc` is equal to 1, `verTapLength[i]` is set equal to `ver_tap_length_minus1[i]` plus 1 for *i* in the range of 0..`num_vertical_filters` – 1.
- Otherwise (`ver_chroma_filter_idc` is equal to 2), the value of `verTapLength[]` is derived as specified in Table D.18.

The variable `verFilterCoeff[i][j]` is derived as follows:

- If `ver_chroma_filter_idc` is equal to 1, `verFilterCoeff[i][j]` is set equal to `ver_filter_coeff[i][j]` for *i* in the range of 0..`num_vertical_filters` – 1, inclusive, and *j* in the range of 0..`ver_tap_length_minus1[i]`, inclusive.
- Otherwise (`ver_chroma_filter_idc` is equal to 2), the values of `verFilterCoeff[][]` are derived as specified in Table D.18.

num_horizontal_filters specifies the number of filters indicated for chroma downsampling and upsampling in the horizontal direction. When `hor_chroma_filter_idc` is equal to 1, depending on the value of `chromaSampleLocType`, the value of `num_horizontal_filters` shall be equal to the values specified in Table D.17.

Table D.17 – Constraints on the value of num_horizontal_filters

chromaSampleLocType	Mandatory value of num_horizontal_filters
0, 2, 4	3
1, 3, 5	2

hor_tap_length_minus1[i] plus 1 specifies the length of the *i*-th filter in the horizontal direction. The value of `hor_tap_length_minus1[i]` shall be in the range of 0 to 31, inclusive.

hor_filter_coeff[i][j] specifies the value of the *j*-th coefficient of the *i*-th filter in the horizontal direction. The value of `hor_filter_coeff[i][j]` shall be in the range of $-2^{31} + 1$ to $2^{31} - 1$, inclusive.

The variable `horTapLength[]` is derived as follows:

- If `hor_chroma_filter_idc` is equal to 1, `horTapLength[i]` is set equal to `hor_tap_length_minus1[i]` plus 1 for *i* in the range of 0 to `num_horizontal_filters` – 1, inclusive.

- Otherwise (hor_chroma_filter_idc is equal to 2), the values of horTapLength[] are derived as specified in Table D.19.

The variable horFilterCoeff[][] is derived as follows:

- If hor_chroma_filter_idc is equal to 1, horFilterCoeff[i][j] is set equal to hor_filter_coeff[i][j] for i in the range of 0 to num_horizontal_filters – 1, inclusive, and j in the range of 0 to hor_tap_length_minus1[i], inclusive.
- Otherwise (hor_chroma_filter_idc is equal to 2), the values of horFilterCoeff[][] are derived as specified in Table D.19.

Table D.18 specifies the coefficients of the sets of chroma downsampling and upsampling filters in the vertical direction when ver_chroma_filter_idc is equal to 2.

NOTE 2 – When ver_chroma_filter_idc is equal to 2, the filter coefficient values specified in Table D.18 correspond to those described in SMPTE RP 2050-1 (2012).

Table D.19 specifies the coefficients of the sets of chroma downsampling and upsampling filters in the horizontal direction when hor_chroma_filter_idc is equal to 2.

NOTE 3 – When hor_chroma_filter_idc is equal to 2, the filter coefficient values specified in Table D.19 correspond to those described in the 5/3 filter specification part of Rec. ITU-T T.800 | ISO/IEC 15444-1.

Table D.18 – Values of verFilterCoeff and verTapLength when ver_chroma_filter_idc is equal to 2

chromaSampleLocType	ver_filtering_field_processing_flag	upsamplingFlag	verFilterCoeff[][]	verTapLength[]
0, 1	0	0	verFilterCoeff[0][] = { -3, -19, 34, 500, 500, 34, -19, -3 }	verTapLength[0] = 8
		1	verFilterCoeff[1][] = { 19, 103, 1037, -135 }	verTapLength[1] = 4
	1	0	verFilterCoeff[0][] = { -8, -26, 115, 586, 409, -48, -4, 0 }	verTapLength[0] = 8
		1	verFilterCoeff[1][] = { 24, -41, 1169, -128 }	verTapLength[1] = 4
			verFilterCoeff[2][] = { -76, 783, 330, -13 }	verTapLength[2] = 4

Table D.19 – Values of horFilterCoeff and horTapLength when hor_chroma_filter_idc is equal to 2

chromaSampleLocType	upsamplingFlag	horFilterCoeff[][]	horTapLength[]
0, 2, 4	0	horFilterCoeff[0][] = { -1, 2, 6, 2, -1 }	horTapLength[0] = 5
	1	horFilterCoeff[1][] = { 1 }	horTapLength[1] = 1
		horFilterCoeff[2][] = { 1, 1 }	horTapLength[2] = 2

The chroma resampling filtering process is applied as follows:

- The variable bottomFlag is derived as follows:
 - If field_seq_flag is equal to 1 and pic_struct is equal to 2, 10 or 12, bottomFlag is set equal to 1.
 - Otherwise, if field_seq_flag is equal to 0 and ver_filtering_field_processing_flag is equal to 1 and the output field being processed is a bottom field, bottomFlag is set equal to 1.
 - Otherwise, bottomFlag is set equal to 0.
- The variables phaseOffsetUp and phaseOffsetDown are derived as follows:
 - If bottomFlag is equal to 1, phaseOffsetUp is set equal to 2 and phaseOffsetDown is set equal to 1.
 - Otherwise (bottomFlag is equal to 0), phaseOffsetUp is set equal to 0 and phaseOffsetDown is set equal to 0.
- The vertical downsampling and upsampling filter coefficients fDv[] and fUv[] are specified in Table D.20.
- The horizontal downsampling and upsampling filter coefficients fDh[] and fUh[] are specified in Table D.21.
- The vertical downsampling and upsampling filter tap lengths lenDv[] and lenUv[] are specified in Table D.20.
- The horizontal downsampling and upsampling filter tap lengths lenDh[] and lenUh[] are specified in Table D.21.
- When chroma_format_idc is equal to 1 and target_format_idc is equal to either 2 or 3, the chroma upsampling filtering process in the vertical direction is applied once on the Cb samples and once on the Cr samples of the decoded cropped output picture as follows:
 - The variables w0 and h0 are derived as follows:

$$\begin{aligned} w0 &= (\text{pic_width_in_luma_samples} / \text{SubWidthC}) - (\text{conf_win_right_offset} + \text{conf_win_left_offset}) \\ h0 &= (\text{pic_height_in_luma_samples} \gg 1) - (\text{conf_win_top_offset} + \text{conf_win_bottom_offset}) \end{aligned} \quad (\text{D-44})$$

- Let p0X[i][j], with i in the range of 0 to w0 – 1, inclusive, j in the range of 0 to h0 – 1, inclusive, and X being either Cb or Cr, be the input array of Cb or Cr chroma samples from the decoded cropped output picture, and p1X[i][j], with i in the range of 0 to w0 – 1, inclusive, j in the range of 0 to (h0 << 1) – 1, inclusive, and X being either Cb or Cr, be the output array of Cb or Cr chroma samples of the vertical chroma upsampling process.

```

divUv[ 0 ] = 0
divUv[ 1 ] = 0
for( j = 0; j < lenUv[ phaseOffsetUp ]; j++ )
    divUv[ 0 ] += fUv[ phaseOffsetUp ][ j ]
for( j = 0; j < lenUv[ 1 + phaseOffsetUp ]; j++ )
    divUv[ 1 ] += fUv[ 1 + phaseOffsetUp ][ j ]
for( u = 0; u < w0; u++ )
    for( v = 0; v < ( h0 << 1 ); v++ ) {
        sum = 0
        posOffsetUp = v % 2 + phaseOffsetUp
        for( j = - ( lenUv[ posOffsetUp ] - 1 ) / 2; j <= lenUv[ posOffsetUp ] / 2; j++ ) {
            sum += p0X[ u ][ Clip3( 0, h0 - 1, ( v >> 1 ) + j ) ]
                * fUv[ posOffsetUp ][ j + ( lenUv[ posOffsetUp ] - 1 ) / 2 ]
        }
        p1X[ u ][ v ] = ( sum + ( divUv[ v % 2 ] >> 1 ) ) / divUv[ v % 2 ]
    }

```

(D-45)

- When ver_filtering_field_process_flag is equal to 1 and field_seq_flag is equal to 0, the chroma upsampling filtering process in the vertical direction is applied to each field of the Cb or Cr chroma components of the cropped output frame picture p0X[i][j], with i in the range of 0 to w0 – 1, inclusive, j in the range of 0 to h0 – 1, inclusive, and X being either Cb or Cr. The following ordered steps apply:
 1. The array p0X is deinterleaved into two fields, p0XTop with height being equal to h0 >> 1 and p0XBottom with height being equal to h0 – (h0 >> 1).
 2. The chroma upsampling filtering process in the vertical direction according to Equation D-45 is applied to p0XTop and p0XBottom to derive the output of the filtering process p1XTop and p1XBottom.
 3. The arrays p1XTop and p1XBottom are interleaved to form the output array p1X of the upsampling filtering process.
- When chroma_format_idc is equal to either 1 or 2 and target_format_idc is equal to 3, the chroma upsampling filtering process in the horizontal direction is applied once on the Cb samples and once on the Cr samples of the decoded picture as follows:

- The variables $w0$ and $h0$ are derived as follows:

$$\begin{aligned} h0 &= (\text{pic_height_in_luma_samples} / \text{SubHeightC}) - (\text{conf_win_top_offset} + \text{conf_win_bottom_offset}) \\ w0 &= (\text{pic_width_in_luma_samples} \gg 1) - (\text{conf_win_right_offset} + \text{conf_win_left_offset}) \end{aligned} \quad (\text{D-46})$$

- Let $p0X[i][j]$, with i in the range of 0 to $w0 - 1$, inclusive, j in the range of 0 to $h0 - 1$, inclusive, and X being either Cb or Cr, be the input array of Cb or Cr chroma samples from the decoded cropped output picture, and $p1X[i][j]$, with i in the range of 0 to $(w0 \ll 1) - 1$, inclusive, j in the range of 0 to $h0 - 1$, inclusive and X being either Cb or Cr, be the output array of Cb or Cr chroma samples of the horizontal chroma upsampling process.

```
divUh[ 0 ] = 0
divUh[ 1 ] = 0
for( j = 0; j < lenUh[ 0 ]; j++ )
    divUh[ 0 ] += fUh[ 0 ][ j ]
for( j = 0; j < lenUh[ 1 ]; j++ )
    divUh[ 1 ] += fUh[ 1 ][ j ]
for( v = 0; v < h0; v++ )
    for( u = 0; u < ( w0 << 1 ); u++ ) {
        sum = 0
        for( i = -( lenUh[ u % 2 ] - 1 ) / 2; i <= lenUh[ u % 2 ] / 2; i++ )
            sum += p0X[ Clip3( 0, w0 - 1, ( u >> 1 ) + i ) ][ v ] * fUh[ u % 2 ][ i + ( lenUh[ u % 2 ] - 1 ) / 2 ]
        p1X[ u ][ v ] = ( sum + ( divUh[ u % 2 ] >> 1 ) ) / divUh[ u % 2 ]
    }
```

(D-47)

- When chroma_format_idc is equal to either 3 or 2 and target_format_idc is equal to 1, the chroma downsampling filtering process in the vertical direction is applied once on the Cb samples and once on the Cr samples of the decoded picture as follows:

- The variables $w0$ and $h0$ are derived as follows:

$$\begin{aligned} w0 &= (\text{pic_width_in_luma_samples} / \text{SubWidthC}) - (\text{conf_win_right_offset} + \text{conf_win_left_offset}) \\ h0 &= \text{pic_height_in_luma_samples} - (\text{conf_win_top_offset} + \text{conf_win_bottom_offset}) \end{aligned} \quad (\text{D-48})$$

- Let $p0X[i][j]$, with i in the range of 0 to $w0 - 1$, inclusive, j in the range of 0 to $h0 - 1$, inclusive, and X being either Cb or Cr, be the input array of Cb or Cr chroma samples from the decoded cropped output picture, and $p1X[i][j]$, with i in the range of 0 to $w0 - 1$, inclusive, j in the range of 0 to $(h0 \gg 1) - 1$, inclusive, and X being either Cb or Cr, be the output array of Cb or Cr chroma samples of the vertical chroma downsampling process.

```
divDv = 0
for( j = 0; j < lenDv[ phaseOffsetDown ]; j++ )
    divDv += fDv[ phaseOffsetDown ][ j ]
for( u = 0; u < w0; u++ )
    for( v = 0; v < ( h0 >> 1 ); v++ ) {
        sum = 0
        for( j = -( lenDv[ phaseOffsetDown ] - 1 ) / 2; j <= lenDv[ phaseOffsetDown ] / 2; j++ )
            sum += p0X[ u ][ Clip3( 0, h0 - 1, ( v << 1 ) + j ) ]
                * fDv[ phaseOffsetDown ][ j + ( lenDv[ phaseOffsetDown ] - 1 ) / 2 ]
        p1X[ u ][ v ] = ( sum + ( divDv >> 1 ) ) / divDv
    }
```

(D-49)

- When $\text{ver_filtering_field_process_flag}$ is equal to 1 and field_seq_flag is equal to 0, the chroma downsampling filtering process in the vertical direction is applied to each field of each of the Cb and Cr chroma components of the cropped output frame picture $p0X[i][j]$, with i in the range of 0 to $w0 - 1$, inclusive, j in the range of 0 to $h0 - 1$, inclusive, and X being either Cb or Cr. The following ordered steps apply:

1. The array $p0X$ is deinterleaved into two fields, $p0X_{\text{Top}}$ with height being equal to $h0 \gg 1$ and $p0X_{\text{Bottom}}$ with height being equal to $h0 - (h0 \gg 1)$.
2. The chroma downsampling filtering process in the vertical direction according to Equation D-49 is applied to $p0X_{\text{Top}}$ and $p0X_{\text{Bottom}}$ to derive the output of the filtering process $p1X_{\text{Top}}$ and $p1X_{\text{Bottom}}$.

3. The arrays p1XTop and p1XBottom are interleaved to form the output array p1X of the downsampling filtering process.

– When chroma_format_idc is equal to 3 and target_format_idc is equal to either 1 or 2, the chroma downsampling filtering process in the horizontal direction is applied once on the Cb samples and once on the Cr samples of the decoded picture as follows:

– The variables w0 and h0 are derived as follows:

$$\begin{aligned} h0 &= (\text{pic_height_in_luma_samples} / \text{SubHeightC}) - (\text{conf_win_top_offset} + \text{conf_win_bottom_offset}) \\ w0 &= \text{pic_width_in_luma_samples} - (\text{conf_win_right_offset} + \text{conf_win_left_offset}) \end{aligned} \quad (\text{D-50})$$

– Let p0X[i][j], with i in the range of 0 to w0 – 1, inclusive, j in the range of 0 to h0 – 1, inclusive, and X being either Cb or Cr, be the input array of Cb or Cr chroma samples from the decoded cropped output picture, and p1X[i][j], with i in the range of 0 to (w0 >> 1) – 1, inclusive, j in the range of 0 to h0 – 1, inclusive, and X being either Cb or Cr, be the output array of Cb or Cr chroma samples of the horizontal chroma downsampling process.

```
divUh = 0
for( j = 0; j < lenDh; j++ )
    divDh += fDh[ 0 ][ j ]
for( v = 0; v < h0; v++ )
    for( u = 0; u < ( w0 >> 1 ); u++ ) {
        sum = 0
        for( i = -( lenDh - 1 ) / 2; i <= lenDh / 2; i++ )
            sum += p0X[ Clip3( 0, w0 - 1, ( u << 1 ) + i ) ][ v ] * fDh[ 0 ][ i + ( lenDh - 1 ) / 2 ]
        p1X[ u ][ v ] = ( sum + ( divDh >> 1 ) ) / divDh
    }
```

(D-51)

Table D.20 – Usage of chroma filter in the vertical direction

chromaSampleLocationType	ver_filtering_field_processing_flag	num_vertical_filters (when applicable)	upsamplingFlag	bottomFlag	Filter coefficients (fDv for downsampling or fUv for upsampling)	Filter tap length (lenDv for downsampling or lenUv for upsampling)
0, 1	0	2	0	–	$fDv[0][j] = \text{verFilterCoeff}[0][j]$ $j = 0..lenDv[0] - 1$	$lenDv[0] = \text{verTapLength}[0]$
			1	–	$fUv[0][j] = \text{verFilterCoeff}[1][j]$ $j = 0..lenUv[0] - 1$	$lenUv[0] = \text{verTapLength}[1]$
					$fUv[1][j] = \text{verFilterCoeff}[1][\text{verTapLength}[1] - j - 1]$ $j = 0..lenUv[1] - 1$	$lenUv[1] = \text{verTapLength}[1]$
	1	3	0	0	$fDv[0][j] = \text{verFilterCoeff}[0][j]$ $j = 0..lenDv[0] - 1$	$lenDv[0] = \text{verTapLength}[0]$
				1	$fDv[1][j] = \text{verFilterCoeff}[0][\text{verTapLength}[0] - j - 1]$ $j = 0..lenDv[1] - 1$	$lenDv[1] = \text{verTapLength}[0]$
			1	0	$fUv[0][j] = \text{verFilterCoeff}[1][j]$ $j = 0..lenUv[0] - 1$	$lenUv[0] = \text{verTapLength}[1]$

Table D.20 – Usage of chroma filter in the vertical direction

chromaSampleLocType	ver_filtering_field_processing_flag	num_vertical_filters (when applicable)	upsamplingFlag	bottomFlag	Filter coefficients (fDv for downsampling or fUv for upsampling)	Filter tap length (lenDv for downsampling or lenUv for upsampling)
					$fUv[1][j] = verFilterCoeff[2][j]$ $j = 0..lenUv[1] - 1$	$lenUv[1] = verTapLength[2]$
				1	$fUv[2][j] = verFilterCoeff[1][verTapLength[1] - j - 1]$ $j = 0..lenUv[2] - 1$	$lenUv[2] = verTapLength[1]$
					$fUv[3][j] = verFilterCoeff[2][verTapLength[2] - j - 1]$ $j = 0..lenUv[3] - 1$	$lenUv[3] = verTapLength[2]$
2, 3	0	3	0	–	$fDv[0][j] = verFilterCoeff[0][j]$ $j = 0..lenDv[0] - 1$	$lenDv[0] = verTapLength[0]$
			1	–	$fUv[0][j] = verFilterCoeff[1][j]$ $j = 0..lenUv[0] - 1$	$lenUv[0] = verTapLength[1]$
					$fUv[1][j] = verFilterCoeff[2][j]$ $j = 0..lenUv[1] - 1$	$lenUv[1] = verTapLength[2]$
	1	5	0	0	$fDv[0][j] = verFilterCoeff[0][j]$ $j = 0..lenDv[0] - 1$	$lenDv[0] = verTapLength[0]$
				1	$fDv[1][j] = verFilterCoeff[1][j]$ $j = 0..lenDv[1] - 1$	$lenDv[1] = verTapLength[1]$
			1	0	$fUv[0][j] = verFilterCoeff[2][j]$ $j = 0..lenUv[0] - 1$	$lenUv[0] = verTapLength[2]$
					$fUv[1][j] = verFilterCoeff[3][j]$ $j = 0..lenUv[1] - 1$	$lenUv[1] = verTapLength[3]$
				1	$fUv[2][j] = verFilterCoeff[4][j]$ $j = 0..lenUv[2] - 1$	$lenUv[2] = verTapLength[4]$
					$fUv[3][j] = verFilterCoeff[4][verTapLength[4] - j - 1]$ $j = 0..lenUv[3] - 1$	$lenUv[3] = verTapLength[4]$
4, 5	0	3	0	–	$fDv[0][j] = verFilterCoeff[0][j]$ $j = 0..lenDv[0] - 1$	$lenDv[0] = verTapLength[0]$
			1	–	$fUv[0][j] = verFilterCoeff[1][j]$ $j = 0..lenUv[0] - 1$	$lenUv[0] = verTapLength[1]$
					$fUv[1][j] = verFilterCoeff[2][j]$ $j = 0..lenUv[1] - 1$	$lenUv[1] = verTapLength[2]$
	1	5	0	0	$fDv[0][j] = verFilterCoeff[0][j]$ $= 0..lenDv[0] - 1$	$lenDv[0] = verTapLength[0]$
				1	$fDv[1][j] = verFilterCoeff[1][j]$ $j = 0..lenDv[1] - 1$	$lenDv[1] = verTapLength[1]$

Table D.20 – Usage of chroma filter in the vertical direction

chromaSampleLocType	ver_filtering_field_processing_flag	num_vertical_filters (when applicable)	upsamplingFlag	bottomFlag	Filter coefficients (FDv for downsampling or fUv for upsampling)	Filter tap length (lenDv for downsampling or lenUv for upsampling)
			1	0	$fUv[0][j] = verFilterCoeff[2][j]$ $j = 0..lenUv[0] - 1$	$lenUv[0] = verTapLength[2]$
					$fUv[1][j] = verFilterCoeff[2][verTapLength[2] - j - 1]$ $j = 0..lenUv[1] - 1$	$lenUv[1] = verTapLength[2]$
				1	$fUv[2][j] = verFilterCoeff[3][j]$ $j = 0..lenUv[2] - 1$	$lenUv[2] = verTapLength[3]$
					$fUv[3][j] = verFilterCoeff[4][j]$ $j = 0..lenUv[3] - 1$	$lenUv[3] = verTapLength[4]$

Table D.21 – Usage of chroma filter in the horizontal direction

chromaSampleLocType	num_horizontal_filters (when applicable)	upsamplingFlag	Filter coefficients (FDh for downsampling or fUh for upsampling)	Filter tap length (lenDh for downsampling or lenUh for upsampling)
0,2,4	3	0	$fDh[0][j] = horFilterCoeff[0][j]$ $j = 0..lenDh - 1$	$lenDh = horTapLength[0]$
		1	$fUh[0][j] = horFilterCoeff[1][j]$ $j = 0..lenUh[0] - 1$	$lenUh[0] = horTapLength[1]$
			$fUh[1][j] = horFilterCoeff[2][j]$ $j = 0..lenUh[1] - 1$	$lenUh[1] = horTapLength[2]$
1,3,5	2	0	$fDh[0][j] = horFilterCoeff[0][j]$ $j = 0..lenDh - 1$	$lenDh = horTapLength[0]$
		1	$fUh[0][j] = horFilterCoeff[1][j]$ $j = 0..lenUh[0] - 1$	$lenUh[0] = horTapLength[1]$
			$fUh[1][j] = horFilterCoeff[1][horTapLength[1] - j - 1]$ $j = 0..lenUh[1] - 1$	$lenUh[1] = horTapLength[1]$

D.3.32 Knee function information SEI message semantics

The knee function information SEI message provides information to enable mapping of the colour samples of decoded pictures for customisation to particular display environments. The process uses a knee function to map the white level of sample values in the normalized linear RGB colour space to an appropriate luminance level and should be applied to each RGB component produced by the colour space conversion of a decoded image.

A knee function is a piece-wise linear function that starts with the point (0.0, 0.0), ends with the point (1.0, 1.0) and is specified by knee points in ascending order of index i . The coordinate of the i -th knee point is specified by ($\text{input_knee_point}[i] \div 1\,000$, $\text{output_knee_point}[i] \div 1\,000$). A knee function maps the input luminance level normalized in the range of 0.0 to 1.0 to the output luminance level normalized in the range of 0.0 to 1.0. An example of a knee function is shown in Figure D.11.

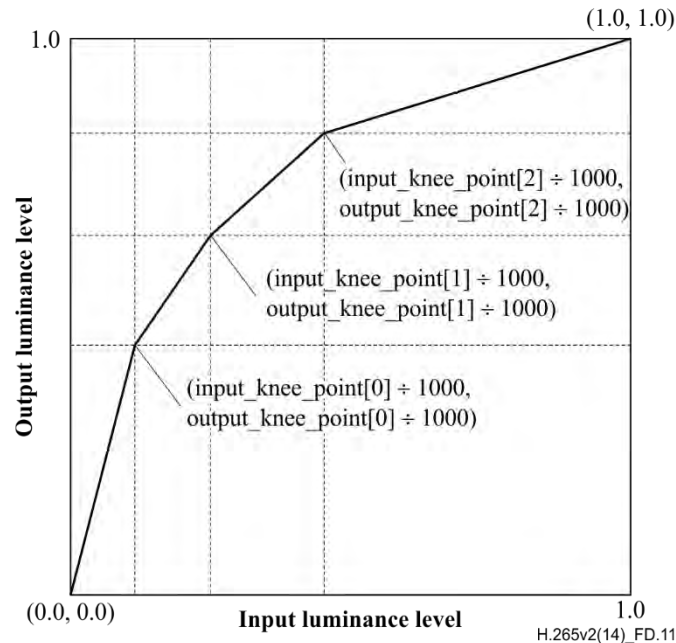


Figure D.11 – A knee function with `num_knee_points_minus1` equal to 2

knee_function_id contains an identifying number that may be used to identify the purpose of the knee functions. The value of `knee_function_id` shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of `knee_function_id` from 0 to 255, inclusive, and from 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of `knee_function_id` from 256 to 511, inclusive, and from 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of `knee_function_id` in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

NOTE 1 – The `knee_function_id` can be used to support knee function processes that are suitable for different display scenarios. For example, different values of `knee_function_id` may correspond to different display bit depths.

knee_function_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous knee function information SEI message in output order that applies to the current layer. `knee_function_cancel_flag` equal to 0 indicates that knee function information follows.

knee_function_persistence_flag specifies the persistence of the knee function information SEI message for the current layer.

`knee_function_persistence_flag` equal to 0 specifies that the knee function information applies to the current decoded picture only.

Let `picA` be the current picture. `knee_function_persistence_flag` equal to 1 specifies that the knee function information persists for the current layer in output order until any of the following conditions are true:

- A new CLVS of the current layer begins.
- The bitstream ends.
- A picture `picB` in the current layer in an access unit containing a knee function information SEI message with the same value of `knee_function_id` and applicable to the current layer is output for which `PicOrderCnt(picB)` is greater

than `PicOrderCnt(picA)`, where `PicOrderCnt(picB)` and `PicOrderCnt(picA)` are the `PicOrderCntVal` values of `picB` and `picA`, respectively, immediately after the invocation of the decoding process for picture order count for `picB`.

input_d_range specifies the peak luminance level for the input picture of the knee function process relative to the nominal luminance level in units of 0.1%. When the value of `input_d_range` is equal to 0, the peak luminance level of the input picture is unspecified.

input_disp_luminance specifies the expected display brightness of peak luminance level for the input picture of the knee function process. The value of `input_disp_luminance` is in units of candelas per square metre. When the value of `input_disp_luminance` is equal to 0, the expected display brightness of peak luminance level for the input picture is unspecified.

output_d_range specifies the peak luminance level for the output picture of the knee function process relative to the nominal luminance level in units of 0.1%. When the value of `output_d_range` is equal to 0, the peak luminance level of the output picture is unspecified.

output_disp_luminance specifies the expected display brightness of peak luminance level for the output picture of the knee function process. The value of `output_disp_luminance` is in units of candelas per square metre. When the value of `output_disp_luminance` is equal to 0, the expected display brightness of peak luminance level for the output picture is unspecified.

num_knee_points_minus1 plus 1 specifies the number of knee points used to define the knee function. `num_knee_points_minus1` shall be in the range of 0 to 998, inclusive.

input_knee_point[i] specifies the luminance level of the *i*-th knee point of the input picture. The luminance level of the knee point of the input picture is normalized to the range of 0 to 1.0 in units of 0.1%. The value of `input_knee_point[i]` shall be in the range of 1 to 999, inclusive. The value of `input_knee_point[i]` shall be greater than the value of `input_knee_point[i - 1]`, for *i* in the range of 1 to `num_knee_points_minus1`, inclusive.

output_knee_point[i] specifies the luminance level of the *i*-th knee of the output picture. The luminance level of the knee point of the output picture is normalized to the range of 0 to 1.0 in units of 0.1%. The value of `output_knee_point[i]` shall be in the range of 0 to 1 000, inclusive. The value of `output_knee_point[i]` shall be greater than or equal to the value of `output_knee_point[i - 1]`, for *i* in the range of 1 to `num_knee_points_minus1`, inclusive.

NOTE 2 – The luminance level conversion process between an input signal *x* and an output signal *y*, where the luminance levels for both input and output are normalized to be in the range of 0.0 to 1.0, is specified as follows:

```

if( x <= input_knee_point[ 0 ] ÷ 1 000 )
    y = ( output_knee_point[ 0 ] ÷ input_knee_point[ 0 ] ) * x
else if( x > input_knee_point[ num_knee_points_minus1 ] )
    y = ( ( 1 000 - output_knee_point[ num_knee_points_minus1 ] ) ÷
          ( 1 000 - input_knee_point[ num_knee_points_minus1 ] ) ) *
          ( x - input_knee_point[ num_knee_points_minus1 ] ÷ 1 000 ) +
          ( output_knee_point[ num_knee_points_minus1 ] ÷ 1 000 )
else
    for( i = 1; i <= num_knee_points_minus1; i++ )
        if( input_knee_point[ i - 1 ] ÷ 1 000 < x && x <= input_knee_point[ i ] ÷ 1 000 )
            y = ( ( output_knee_point[ i ] - output_knee_point[ i - 1 ] ) ÷
                  ( input_knee_point[ i ] - input_knee_point[ i - 1 ] ) ) *
                  ( x - input_knee_point[ i - 1 ] ÷ 1 000 ) + ( output_knee_point[ i - 1 ] ÷ 1 000 )

```

(D-52)

D.3.33 Colour remapping information SEI message semantics

The colour remapping information SEI message provides information to enable remapping of the reconstructed colour samples of the output pictures for purposes such as converting the output pictures to a representation that is more suitable for an alternative display. The colour remapping model used in the colour remapping information SEI message is composed of a first piece-wise linear function applied to each colour component (specified by the "pre" set of syntax elements herein), followed by a three-by-three matrix applied to the three resulting colour components, followed by a second piece-wise linear function applied to each resulting colour component (specified by the "post" set of syntax elements herein).

NOTE 1 – Colour remapping of the output pictures for the display process (which is outside the scope of this Specification) is optional and does not affect the decoding process specified in this Specification.

Unless indicated otherwise by some means not specified in this Specification, the input to the indicated remapping process is the set of decoded sample values after applying an (unspecified) upsampling conversion process to the 4:4:4 colour sampling format as necessary when the colour remapping three-by-three matrix coefficients are present in the SEI message and `chroma_format_idc` is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format). When `chroma_format_idc` is equal to 0 (monochrome), the colour remapping information SEI message shall not be present, although decoders shall allow such messages to be present and shall ignore any such colour remapping information SEI messages that may be present.

colour_remap_id contains an identifying number that may be used to identify the purpose of the colour remapping information. The value of colour_remap_id may be used (in a manner not specified in this Specification) to indicate that the input to the remapping process is the output of some conversion process that is not specified in this Specification, such as a conversion of the picture to some alternative colour representation (e.g., conversion from a YCbCr colour representation to a GBR colour representation). When more than one colour remapping information SEI message is present with the same value of colour_remap_id, the content of these colour remapping information SEI messages shall be the same. When colour remapping information SEI messages are present that have more than one value of colour_remap_id, this may indicate that the remapping processes indicated by the different values of colour_remap_id are alternatives that are provided for different purposes or that a cascading of remapping processes is to be applied in a sequential order (an order that is not specified in this Specification). The value of colour_remap_id shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of colour_remap_id from 0 to 255, inclusive, and from 512 to $2^{31} - 1$, inclusive, may be used as determined by the application. Values of colour_remap_id from 256 to 511, inclusive, and from 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of colour_remap_id in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

NOTE 2 – The colour_remap_id can be used to support different colour remapping processes that are suitable for different display scenarios. For example, different values of colour_remap_id may correspond to different remapped colour spaces supported by displays.

colour_remap_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous colour remapping information SEI message in output order that applies to the current layer. colour_remap_cancel_flag equal to 0 indicates that colour remapping information follows.

colour_remap_persistence_flag specifies the persistence of the colour remapping information SEI message for the current layer.

colour_remap_persistence_flag equal to 0 specifies that the colour remapping information applies to the current picture only.

Let picA be the current picture. colour_remap_persistence_flag equal to 1 specifies that the colour remapping information persists for the current layer in output order until either of the following conditions is true:

- A new CLVS of the current layer begins.
- The bitstream ends.
- A picture picB in the current layer in an access unit containing a colour remapping information SEI message with the same value of colour_remap_id and applicable to the current layer is output for which PicOrderCnt(picB) is greater than PicOrderCnt(picA), where PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

colour_remap_video_signal_info_present_flag equal to 1 specifies that syntax elements colour_remap_full_range_flag, colour_remap_primaries, colour_remap_transfer_function and colour_remap_matrix_coefficients are present, colour_remap_video_signal_info_present_flag equal to 0 specifies that syntax elements colour_remap_full_range_flag, colour_remap_primaries, colour_remap_transfer_function and colour_remap_matrix_coefficients are not present.

colour_remap_full_range_flag has the same semantics as specified in clause E.3.1 for the video_full_range_flag syntax element, except that colour_remap_full_range_flag identifies the colour space of the remapped reconstructed picture, rather than the colour space used for the CLVS. When not present, the value of colour_remap_full_range_flag is inferred to be equal to the value of video_full_range_flag.

colour_remap_primaries has the same semantics as specified in clause E.3.1 for the colour_primaries syntax element, except that colour_remap_primaries identifies the colour space of the remapped reconstructed picture, rather than the colour space used for the CLVS. When not present, the value of colour_remap_primaries is inferred to be equal to the value of colour_primaries.

colour_remap_transfer_function has the same semantics as specified in clause E.3.1 for the transfer_characteristics syntax element, except that colour_remap_transfer_function identifies the colour space of the remapped reconstructed picture, rather than the colour space used for the CLVS. When not present, the value of colour_remap_transfer_function is inferred to be equal to the value of transfer_characteristics.

colour_remap_matrix_coefficients has the same semantics as specified in clause E.3.1 for the matrix_coeffs syntax element, except that colour_remap_matrix_coefficients identifies the colour space of the remapped reconstructed picture, rather than the colour space used for the CLVS. When not present, the value of colour_remap_matrix_coefficients is inferred to be equal to the value of matrix_coeffs.

colour_remap_input_bit_depth specifies the bit depth of the colour components of the associated pictures for purposes of interpretation of the colour remapping information SEI message. When any colour remapping information SEI message is present with the value of colour_remap_input_bit_depth not equal to the bit depth of the decoded colour components,

the SEI message refers to the hypothetical result of a conversion operation performed to convert the decoded colour component samples to the bit depth equal to colour_remap_input_bit_depth.

The value of colour_remap_input_bit_depth shall be in the range of 8 to 16, inclusive. Values of colour_remap_input_bit_depth from 0 to 7, inclusive, and from 17 to 255, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore all colour remapping SEI messages that contain a colour_remap_input_bit_depth in the range of 0 to 7, inclusive, or in the range of 17 to 255, inclusive, and bitstreams shall not contain such values.

colour_remap_output_bit_depth specifies the bit depth of the output of the colour remapping function described by the colour remapping information SEI message.

The value of colour_remap_output_bit_depth shall be in the range of 8 to 16, inclusive. Values of colour_remap_output_bit_depth from 0 to 7, inclusive, and in the range of 17 to 255, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore all colour remapping SEI messages that contain a value of colour_remap_output_bit_depth from 0 to 7, inclusive, or in the range of 17 to 255, inclusive, and bitstreams shall not contain such values.

pre_lut_num_val_minus1[c] plus 1 specifies the number of pivot points in the piece-wise linear remapping function for the c-th component, where c equal to 0 refers to the luma or G component, c equal to 1 refers to the Cb or B component and c equal to 2 refers to the Cr or R component. When pre_lut_num_val_minus1[c] is equal to 0, the default end points of the input values are 0 and $2^{\text{colour_remap_input_bit_depth}} - 1$, and the corresponding default end points of the output values are 0 and $2^{\text{colour_remap_output_bit_depth}} - 1$, for the c-th component. In bitstreams conforming to this version of this Specification, the value of pre_lut_num_val_minus1[c] shall be in the range of 0 to 32, inclusive.

pre_lut_coded_value[c][i] specifies the input value of the i-th pivot point for the c-th component. The number of bits used to represent pre_lut_coded_value[c][i] is $((\text{colour_remap_input_bit_depth} + 7) \gg 3) \ll 3$.

pre_lut_target_value[c][i] specifies the output value of the i-th pivot point for the c-th component. The number of bits used to represent pre_lut_target_value[c][i] is $((\text{colour_remap_output_bit_depth} + 7) \gg 3) \ll 3$.

When pre_lut_coded_value[c][0] is greater than 0, an initial linear segment should be inferred that maps input values ranging from 0 to pre_lut_coded_value[c][0], inclusive, to target values ranging from 0 to pre_lut_target_value[c][0], inclusive.

When pre_lut_coded_value[c][pre_lut_num_val_minus1[c]] is not equal to $2^{\text{colour_remap_input_bit_depth}} - 1$, a final linear segment should be inferred that maps input values ranging from pre_lut_coded_value[c][pre_lut_num_val_minus1[c]] to $2^{\text{colour_remap_input_bit_depth}} - 1$, inclusive, to target values ranging from pre_lut_target_value[c][pre_lut_num_val_minus1[c]] to $2^{\text{colour_remap_output_bit_depth}} - 1$, inclusive.

colour_remap_matrix_present_flag equal to 1 indicates that the syntax elements log2_matrix_denom and colour_remap_coeffs[c][i], for c and i in the range of 0 to 2, inclusive, are present. colour_remap_matrix_present_flag equal to 0 indicates that the syntax elements log2_matrix_denom and colour_remap_coeffs[c][i], for c and i in the range of 0 to 2, inclusive, are not present.

log2_matrix_denom specifies the base 2 logarithm of the denominator for all matrix coefficients. The value of log2_matrix_denom shall be in the range of 0 to 15, inclusive. When not present, the value of log2_matrix_denom is inferred to be equal to 0.

colour_remap_coeffs[c][i] specifies the value of the three-by-three colour remapping matrix coefficients. The value of colour_remap_coeffs[c][i] shall be in the range of -2^{15} to $2^{15} - 1$, inclusive. When colour_remap_coeffs[c][i] is not present, it is inferred to be equal to 1 if c is equal to i, and inferred to be equal to 0 otherwise.

NOTE 3 – When colour_remap_matrix_present_flag is equal to 0, the colour remapping matrix is inferred to be equal to the identity matrix of size 3x3.

The variable matrixOutput[c] for c = 0, 1 and 2 is derived as follows:

```
roundingOffset = log2_matrix_denom == 0 ? 0 : 1 << ( log2_matrix_denom - 1 )
matrixOutput[ c ] = Clip3( 0, ( 1 << colour_remap_output_bit_depth ) - 1,
    ( colour_remap_coeffs[ c ][ 0 ] * matrixInput[ 0 ] + colour_remap_coeffs[ c ][ 1 ] * matrixInput[ 1 ]
    + colour_remap_coeffs[ c ][ 2 ] * matrixInput[ 2 ] + roundingOffset ) >> log2_matrix_denom )
(D-53)
```

where matrixInput[c] is the input sample value of the c-th colour component and matrixOutput[c] is the output sample value of the c-th colour component.

post_lut_num_val_minus1[c] has the same semantics as pre_lut_num_val_minus1[c], with "pre" replaced by "post", except that the default end points of the input values are 0 and $2^{\text{colour_remap_output_bit_depth}} - 1$ for the c-th colour component. The value of post_lut_num_val_minus1[c] shall be in the range of 0 to 32, inclusive.

post_lut_coded_value[c][i] has the same semantics as **pre_lut_coded_value**[c][i], with "pre" replaced by "post", except that the number of bits used to represent **post_lut_coded_value**[c][i] is $((\text{colour_remap_output_bit_depth} + 7) \gg 3) \ll 3$.

post_lut_target_value[c][i] has the same semantics as **pre_lut_target_value**[c][i], with "pre" replaced by "post" except that **colour_remap_input_bit_depth** is replaced by **colour_remap_output_bit_depth** in the semantics.

D.3.34 Deinterlaced field identification SEI message semantics

The deinterlaced picture information SEI message indicates that the current picture represents a frame that was interpolated via a deinterlacing process prior to encoding, and indicates the field parity of the associated source field prior to the deinterlacing process. When a progressive-to-interlace conversion process is applied to the decoded picture prior to display, it is recommended that the field of the decoded frame with the indicated field parity should be used.

When the value of **field_seq_flag** of any active SPS for any picture in the current access unit is equal to 1, the deinterlaced field identification SEI message shall not be present.

deinterlaced_picture_source_parity_flag equal to 0 indicates that the current picture was deinterlaced using a top field picture as the associated source field. **deinterlaced_picture_source_parity_flag** equal to 1 indicates that the current picture was deinterlaced using a bottom field picture as the associated source field.

D.3.35 Content light level information SEI message semantics

This SEI message identifies upper bounds for the nominal target brightness light level of the pictures of the CLVS.

The information conveyed in this SEI message is intended to be adequate for purposes corresponding to the use of the Consumer Electronics Association 861.3 specification.

The semantics of the content light level information SEI message are defined in relation to the values of samples in a 4:4:4 representation of red, green, and blue colour primary intensities in the linear light domain for the pictures of the CLVS, in units of candelas per square metre. However, this SEI message does not, by itself, identify a conversion process for converting the sample values of a decoded picture to the samples in a 4:4:4 representation of red, green, and blue colour primary intensities in the linear light domain for the picture.

NOTE 1 – Other syntax elements, such as **colour primaries**, **transfer characteristics**, **matrix coeffs**, and the chroma resampling filter hint SEI message, when present, may assist in the identification of such a conversion process.

Given the red, green, and blue colour primary intensities in the linear light domain for the location of a luma sample in a corresponding 4:4:4 representation, denoted as E_R , E_G , and E_B , the maximum component intensity is defined as $E_{\text{Max}} = \text{Max}(E_R, \text{Max}(E_G, E_B))$. The light level corresponding to the stimulus is then defined as the CIE 1931 luminance corresponding to equal amplitudes of E_{Max} for all three colour primary intensities for red, green, and blue (with appropriate scaling to reflect the nominal luminance level associated with peak white – e.g., ordinarily scaling to associate peak white with 10 000 candelas per square metre when **transfer_characteristics** is equal to 16).

NOTE 2 – Since the maximum value E_{Max} is used in this definition at each sample location, rather than a direct conversion from E_R , E_G , and E_B to the corresponding CIE 1931 luminance, the CIE 1931 luminance at a location may in some cases be less than the indicated light level. This situation would occur, for example, when E_R and E_G are very small and E_B is large, in which case the indicated light level would be much larger than the true CIE 1931 luminance associated with the (E_R, E_G, E_B) triplet.

All content light level information SEI messages that apply to the same CLVS shall have the same content.

max_content_light_level, when not equal to 0, indicates an upper bound on the maximum light level among all individual samples in a 4:4:4 representation of red, green, and blue colour primary intensities (in the linear light domain) for the pictures of the CLVS, in units of candelas per square metre. When equal to 0, no such upper bound is indicated by **max_content_light_level**.

max_pic_average_light_level, when not equal to 0, indicates an upper bound on the maximum average light level among the samples in a 4:4:4 representation of red, green, and blue colour primary intensities (in the linear light domain) for any individual picture of the CLVS, in units of candelas per square metre. When equal to 0, no such upper bound is indicated by **max_pic_average_light_level**.

NOTE 3 – When the visually relevant region does not correspond to the entire cropped decoded picture, such as for "letterbox" encoding of video content with a wide picture aspect ratio within a taller cropped decoded picture, the indicated average should be performed only within the visually relevant region.

D.3.36 Dependent random access point indication SEI message semantics

The picture associated with a dependent random access point indication SEI message is referred to as a DRAP picture.

The presence of the dependent random access point indication SEI message indicates that the constraints on picture order and picture referencing specified in this clause apply. These constraints can enable a decoder to properly decode the DRAP

picture and the pictures that follow it in both decoding order and output order without needing to decode any other pictures except the associated IRAP picture.

The constraints indicated by the presence of the dependent random access point indication SEI message are as follows:

- The DRAP picture shall be a TRAIL_R picture with TemporalId equal to 0 and nuh_layer_id equal to 0.
- The DRAP picture shall not include any pictures in its RPS lists RefPicSetStCurrBefore, RefPicSetStCurrAfter, and RefPicSetLtCurr except its associated IRAP picture.
- Any picture that follows the DRAP picture in both decoding order and output order shall not include, in its RPS, any picture that precedes the DRAP picture in decoding order or output order with the exception of the IRAP picture associated with the DRAP picture.

D.3.37 Coded region completion SEI message semantics

The coded region completion SEI message indicates the value of the slice_segment_address of the next slice segment in the bitstream (when present) and whether the next slice segment in the bitstream (when present) is an independent slice segment or a dependent slice segment. The term "next slice segment", here and below in this clause, refers to the next slice segment, in decoding order, after the VCL NAL unit associated with the SEI NAL unit containing the coded region completion SEI message.

NOTE – The coded region completion SEI message may be used in determining that a complete slice, coded picture, or access unit has been received prior to receiving any VCL NAL unit of the next slice, coded picture, or access unit, respectively. Consequently, when an implementation of the decoding process expects a complete slice, coded picture, or access unit as input, the coded region completion SEI message may reduce the decoding latency.

next_segment_address identifies the value of the slice_segment_address of the next slice segment in the bitstream (when present). When the next slice segment has first_slice_segment_in_pic_flag equal to 1 or no subsequent slice segment is present in the bitstream, the value of next_segment_address shall be equal to 0.

independent_slice_segment_flag equal to 1 indicates that the next slice segment, when present, is an independent slice segment. independent_slice_segment_flag equal to 0 indicates that the next slice segment, when present, is a dependent slice segment. When independent_slice_segment_flag is not present, it is inferred to be equal to 1.

D.3.38 Alternative transfer characteristics SEI message semantics

The alternative transfer characteristics SEI message provides a preferred alternative value for the transfer_characteristics syntax element that is indicated by the colour description syntax of VUI parameters of the SPS. This SEI message is intended to be used in cases when some value of transfer_characteristics is preferred for interpretation of the pictures of the CLVS although some other value of transfer_characteristics may also be acceptable for interpretation of the pictures of the CLVS and that other value is provided in the colour description syntax of VUI parameters of the SPS for interpretation by decoders that do not support interpretation of the preferred value (e.g., because the preferred value had not yet been defined in a previous version of this Specification).

When an alternative transfer characteristics SEI message is present for any picture of a CLVS of a particular layer and the first picture of the CLVS is an IRAP picture, an alternative transfer characteristics SEI message shall be present for that IRAP picture. The alternative transfer characteristics SEI message persists for the current layer in decoding order from the current picture until the end of the CLVS. All alternative transfer characteristics SEI messages that apply to the same CLVS shall have the same content.

preferred_transfer_characteristics specifies a preferred alternative value for the transfer_characteristics syntax element of the colour description syntax of VUI parameters of the SPS. The semantics for preferred_transfer_characteristics are otherwise the same as for the transfer_characteristics syntax element specified in the VUI parameters of the SPS (see clause E.3.1 and Table E.4). When preferred_transfer_characteristics is not equal to the value of transfer_characteristics indicated in the VUI parameters of the SPS, decoders should ignore the value of transfer_characteristics indicated in the VUI parameters of the SPS and instead use the value indicated by preferred_transfer_characteristics.

D.3.39 Ambient viewing environment SEI message semantics

The ambient viewing environment SEI message identifies the characteristics of the nominal ambient viewing environment for the display of the associated video content. The syntax elements of the ambient viewing environment SEI message may assist the receiving system in adapting the received video content for local display in viewing environments that may be similar or may substantially differ from those assumed or intended when mastering the video content.

This SEI message does not provide information on colour transformations that would be appropriate to preserve creative intent on displays with colour volumes different from that of the described mastering display.

When an ambient viewing environment SEI message is present for any picture of a CLVS of a particular layer and the first picture of the CLVS is an IRAP picture, an ambient viewing environment SEI message shall be present for that IRAP

picture. The ambient viewing environment SEI message persists for the current layer in decoding order from the current picture until the end of the CLVS. All ambient viewing environment SEI messages that apply to the same CLVS shall have the same content.

ambient_illuminance specifies the environmental illuminance of the ambient viewing environment in units of 0.0001 lux. **ambient_illuminance** shall not be equal to 0.

ambient_light_x and **ambient_light_y** specify the normalized x and y chromaticity coordinates, respectively, of the environmental ambient light in the nominal viewing environment, according to the CIE 1931 definition of x and y as specified in ISO 11664-1 (see also ISO 11664-3 and CIE 15), in normalized increments of 0.00002. The values of **ambient_light_x** and **ambient_light_y** shall be in the range of 0 to 50 000, inclusive.

NOTE – For example, the conditions identified in Rec. ITU-R BT.2035 can be expressed using **ambient_illuminance** equal to 100 000 with background chromaticity indicating D_{65} (**ambient_light_x** equal to 15 635, **ambient_light_y** equal to 16 450), or optionally in some regions, background chromaticity indicating D_{93} (**ambient_light_x** equal to 14 155, **ambient_light_y** equal to 14 855).

D.3.40 Content colour volume SEI message semantics

The content colour volume SEI message describes the colour volume characteristics of the associated pictures. These colour volume characteristics are expressed in terms of a nominal range, although deviations from this range may occur.

The variable **transferCharacteristics** is specified as follows:

- If an alternative transfer characteristics SEI message is present for the CLVS, **transferCharacteristics** is set equal to **preferred_transfer_characteristics**;
- Otherwise, (an alternative transfer characteristics SEI message is not present for the CLVS), **transferCharacteristics** is set equal to **transfer_characteristics**.

The content colour volume SEI message shall not be present, and decoders shall ignore it, when any of the following conditions is true:

- Any of the values of **transferCharacteristics**, **colour_primaries**, and **matrix_coeffs** has a value defined as unspecified.
- The value of **transfer_characteristics** is equal to 2, 4, or 5.
- The value of **colour_primaries** is equal to 2.

The following applies when converting the signal from a non-linear to a linear representation:

- If the value of **transferCharacteristics** is equal to 1, 6, 7, 14, or 15, the Rec. ITU-R BT.1886-0 reference electro-optical transfer function should be used to convert the signal to its linear representation, where the value of screen luminance for white is set equal to 100 cd/m², the value of screen luminance for black is set equal to 0 cd/m², and the value of the exponent of the power function is set equal to 2.4.
- Otherwise, if the value of **transferCharacteristics** is equal to 18, the hybrid log-gamma reference electro-optical transfer function specified in Rec. ITU-R BT.2100-2 should be used to convert the signal to its linear representation, where the value of nominal peak luminance of the display is set equal to 1 000 cd/m², the value of the display luminance for black is set equal to 0 cd/m², and the value of system gamma is set equal to 1.2.
- Otherwise (the value of **transferCharacteristics** is not equal to 1, 6, 7, 14, 15, or 18) when the content colour volume SEI message is present, the exact inverse of the transfer function specified in Table E.4 should be used to convert the non-linear signal to a linear representation.

ccv_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous content colour volume SEI message in output order that applies to the current layer. **ccv_cancel_flag** equal to 0 indicates that content colour volume information follows.

ccv_persistence_flag specifies the persistence of the content colour volume SEI message for the current layer.

ccv_persistence_flag equal to 0 specifies that the content colour volume applies to the current decoded picture only.

Let **picA** be the current picture. **ccv_persistence_flag** equal to 1 specifies that the content colour volume SEI message persists for the current layer in output order until any of the following conditions are true:

- A new CLVS of the current layer begins.
- The bitstream ends.
- A picture **picB** in the current layer in an access unit containing a content colour volume SEI message that is applicable to the current layer is output for which **PicOrderCnt(picB)** is greater than **PicOrderCnt(picA)**, where

PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for the picture order count of picB.

ccv_primaries_present_flag equal to 1 specifies that the syntax elements **ccv_primaries_x[c]** and **ccv_primaries_y[c]** are present. **ccv_primaries_present_flag** equal to 0 specifies that the syntax elements **ccv_primaries_x[c]** and **ccv_primaries_y[c]** are not present.

ccv_min_luminance_value_present_flag equal to 1 specifies that the syntax element **ccv_min_luminance_value** is present. **ccv_min_luminance_value_present_flag** equal to 0 specifies that the syntax element **ccv_min_luminance_value** is not present.

ccv_max_luminance_value_present_flag equal to 1 specifies that the syntax element **ccv_max_luminance_value** is present. **ccv_max_luminance_value_present_flag** equal to 0 specifies that the syntax element **ccv_max_luminance_value** is not present.

ccv_avg_luminance_value_present_flag equal to 1 specifies that the syntax element **ccv_avg_luminance_value** is present. **ccv_avg_luminance_value_present_flag** equal to 0 specifies that the syntax element **ccv_avg_luminance_value** is not present.

It is a requirement of bitstream conformance that the values of **ccv_primaries_present_flag**, **ccv_min_luminance_value_present_flag**, **ccv_max_luminance_value_present_flag**, and **ccv_avg_luminance_value_present_flag** shall not all be equal to 0.

ccv_reserved_zero_2bits[i] shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for **reserved_zero_2bits[i]** are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of **reserved_zero_2bits[i]**.

ccv_primaries_x[c] and **ccv_primaries_y[c]** specify the normalized x and y chromaticity coordinates, respectively, of the colour primary component c of the nominal content colour volume, according to the CIE 1931 definition of x and y as specified in ISO 11664-1 (see also ISO 11664-3 and CIE 15), in normalized increments of 0.00002. For describing colour volumes that use red, green, and blue colour primaries, it is suggested that index value c equal to 0 should correspond to the green primary, c equal to 1 should correspond to the blue primary, and c equal to 2 should correspond to the red colour primary (see also Annex E and Table E.3).

The values of **ccv_primaries_x[c]** and **ccv_primaries_y[c]** shall be in the range of –5 000 000 to 5 000 000, inclusive.

When **ccv_primaries_x[c]** and **ccv_primaries_y[c]** are not present, they are inferred to be equal to the normalized x and y chromaticity coordinates, respectively, specified by **colour_primaries**.

ccv_min_luminance_value specifies the normalized minimum luminance value, according to CIE 1931, that is expected to be present in the content, where values are normalized to L_0 or L_c as specified in Table E.4 according to the indicated transfer characteristics of the signal. The values of **ccv_min_luminance_value** are in normalized increments of 0.0000001.

ccv_max_luminance_value specifies the maximum luminance value, according to CIE 1931, that is expected to be present in the content, where values are normalized to L_0 or L_c as specified in Table E.4 according to the transfer characteristics of the signal. The values of **ccv_max_luminance_value** are in normalized increments of 0.0000001.

ccv_avg_luminance_value specifies the average luminance value, according to CIE 1931, that is expected to be present in the content, where values are normalized to L_0 or L_c as specified in Table E.4 according to the transfer characteristics of the signal. The values of **ccv_avg_luminance_value** are in normalized increments of 0.0000001.

NOTE – The resulting domain from this conversion process may or may not represent light in a source or display domain – it is merely a gamut representation domain rather than necessarily being a representation of actual light in either the scene or display domain. Therefore, the values corresponding to **ccv_min_luminance_value**, **ccv_max_luminance_value**, and **ccv_avg_luminance_value** might not necessarily correspond to a true luminance value.

The value of **ccv_min_luminance_value**, when present, shall be less than or equal to **ccv_avg_luminance_value**, when present. The value of **ccv_avg_luminance_value**, when present, shall be less than or equal to **ccv_max_luminance_value**, when present. The value of **ccv_min_luminance_value**, when present, shall be less than or equal to **ccv_max_luminance_value**, when present.

When the visually relevant region does not correspond to the entire cropped decoded picture, such as for "letterbox" encoding of video content with a wide picture aspect ratio within a taller cropped decoded picture, the indicated **ccv_min_luminance_value**, **ccv_max_luminance_value**, and **ccv_avg_luminance_value** should correspond only to values within the visually relevant region.

D.3.41 Semantics of omnidirectional video specific SEI messages

D.3.41.1 Equirectangular projection SEI message semantics

The equirectangular projection SEI message provides information to enable remapping (through an equirectangular projection) of the colour samples of the projected pictures onto a sphere coordinate space in sphere coordinates (ϕ , θ) for use in omnidirectional video applications for which the viewing perspective is from the origin looking outward toward the inside of the sphere. The sphere coordinates are defined so that ϕ is the azimuth (longitude, increasing eastward) and θ is the elevation (latitude, increasing northward).

When an equirectangular projection SEI message is present for any picture of a CLVS of a particular layer, an equirectangular projection SEI message shall be present for the first picture of the CLVS and no SEI message indicating a different type of projection shall be present for any picture of the CLVS.

When `general_non_packed_constraint_flag` is equal to 1 in the active SPS for the current layer, there shall be no equirectangular projection SEI messages applicable for any picture of the CLVS of the current layer.

When `aspect_ratio_idc` is present and greater than 1 in the active SPS for the current layer, there should be no equirectangular projection SEI messages applicable for any picture of the CLVS of the current layer.

A frame packing arrangement SEI message for which all the following conditions are true is referred to as an effectively applicable frame packing arrangement SEI message:

- The value of `frame_packing_arrangement_cancel_flag` is equal to 0.
- The value of `frame_packing_arrangement_type` is equal to 3, 4, or 5.
- The value of `quincunx_sampling_flag` is equal to 0.
- The value of `spatial_flipping_flag` is equal to 0.
- The value of `field_views_flag` is equal to 0.
- The value of `frame0_grid_position_x` is equal to 0.
- The value of `frame0_grid_position_y` is equal to 0.
- The value of `frame1_grid_position_x` is equal to 0.
- The value of `frame1_grid_position_y` is equal to 0.

When a frame packing arrangement SEI message with `frame_packing_arrangement_cancel_flag` equal to 0 that applies to the picture is present that is not an effectively applicable frame packing arrangement SEI message, an equirectangular projection SEI message with `erp_cancel_flag` equal to 0 that applies to the picture shall not be present. Decoders shall ignore equirectangular projection SEI messages when a frame packing arrangement SEI message with `frame_packing_arrangement_cancel_flag` equal to 0 that applies to the picture is present that is not an effectively applicable frame packing arrangement SEI message.

When a segmented rectangular frame packing arrangement SEI message with `segmented_rect_frame_packing_arrangement_cancel_flag` equal to 0 is present that applies to the picture, an equirectangular projection SEI message with `erp_cancel_flag` equal to 0 shall not be present that applies to the picture. Decoders shall ignore equirectangular projection SEI messages when a segmented rectangular frame packing arrangement SEI message with `segmented_rect_frame_packing_arrangement_cancel_flag` equal to 0 is present that applies to the picture.

erp_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous equirectangular projection SEI message in output order. `erp_cancel_flag` equal to 0 indicates that equirectangular projection information follows.

erp_persistence_flag specifies the persistence of the equirectangular projection SEI message for the current layer.

`erp_persistence_flag` equal to 0 specifies that the equirectangular projection SEI message applies to the current decoded picture only.

Let `picA` be the current picture. `erp_persistence_flag` equal to 1 specifies that the equirectangular projection SEI message persists for the current layer in output order until one or more of the following conditions are true:

- A new CLVS of the current layer begins.
- The bitstream ends.
- A picture `picB` in the current layer in an access unit containing an equirectangular projection SEI message that is applicable to the current layer is output for which `PicOrderCnt(picB)` is greater than `PicOrderCnt(picA)`, where

PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

erp_guard_band_flag equal to 1 indicates that the constituent picture contains guard band areas for which the sizes are specified by the syntax elements **erp_left_guard_band_width** and **erp_right_guard_band_width**. **erp_guard_band_flag** equal to 0 indicates that the constituent picture does not contain guard band areas for which the sizes are specified by the syntax elements **erp_left_guard_band_width** and **erp_right_guard_band_width**.

erp_reserved_zero_2bits shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for **erp_reserved_zero_2bits** are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of **erp_reserved_zero_2bits**.

erp_guard_band_type indicates the type of the guard bands as follows:

- **erp_guard_band_type** equal to 0 indicates that the content of the guard band in relation to the content of the constituent picture is unspecified.
- **erp_guard_band_type** equal to 1 indicates that the content of the guard band suffices for interpolation of sample values at sub-pel sample fractional locations within the constituent picture.

NOTE – **erp_guard_band_type** equal to 1 could be used when the boundary samples of a constituent picture have been copied horizontally to the guard band.

- **erp_guard_band_type** equal to 2 indicates that the content of the guard band represents actual picture content at a quality that gradually changes from the picture quality of the constituent picture to that of the spherically adjacent region.
- **erp_guard_band_type** equal to 3 indicates that the content of the guard bands represents actual picture content at a similar level of quality as the constituent picture.
- **erp_guard_band_type** values greater than 3 are reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value of **erp_guard_band_type** when the value is greater than 3 as equivalent to the value 0.

erp_left_guard_band_width specifies the width of the guard band on the left side of the constituent picture in units of luma samples. When **erp_guard_band_flag** is equal to 0, the value of **erp_left_guard_band_width** is inferred to be equal to 0. When **chroma_format_idc** is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format), **erp_left_guard_band_width** shall be an even number.

erp_right_guard_band_width specifies the width of the guard band on the right side of the constituent picture in units of luma samples. When **erp_guard_band_flag** is equal to 0, the value of **erp_right_guard_band_width** is inferred to be equal to 0. When **chroma_format_idc** is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format), **erp_right_guard_band_width** shall be an even number.

D.3.41.2 Cubemap projection SEI message semantics

The cubemap projection SEI message provides information to enable remapping (through a cubemap projection) of the colour samples of the projected pictures onto a sphere coordinate space in sphere coordinates (ϕ , θ) for use in omnidirectional video applications for which the viewing perspective is from the origin looking outward toward the inside of the sphere. The sphere coordinates are defined so that ϕ is the azimuth (longitude, increasing eastward) and θ is the elevation (latitude, increasing northward).

When a cubemap projection SEI message is present for any picture of a CLVS of a particular layer, a cubemap projection SEI message shall be present for the first picture of the CLVS and no SEI message indicating a different type of projection shall be present for any picture of the CLVS.

When **general_non_packed_constraint_flag** is equal to 1 in the active SPS for the current layer, there shall be no cubemap projection SEI messages applicable for any picture of the CLVS of the current layer.

When **aspect_ratio_idc** is present and greater than 1 in the active SPS for the current layer, there should be no cubemap projection SEI messages applicable for any picture of the CLVS of the current layer.

A frame packing arrangement SEI message for which all the following conditions are true is referred to as an effectively applicable frame packing arrangement SEI message:

- The value of **frame_packing_arrangement_cancel_flag** is equal to 0.
- The value of **frame_packing_arrangement_type** is equal to 3, 4, or 5.
- The value of **quincunx_sampling_flag** is equal to 0.
- The value of **spatial_flipping_flag** is equal to 0.

- The value of `field_views_flag` is equal to 0.
- The value of `frame0_grid_position_x` is equal to 0.
- The value of `frame0_grid_position_y` is equal to 0.
- The value of `frame1_grid_position_x` is equal to 0.
- The value of `frame1_grid_position_y` is equal to 0.

When a frame packing arrangement SEI message with `frame_packing_arrangement_cancel_flag` equal to 0 that applies to the picture is present that is not an effectively applicable frame packing arrangement SEI message, a cubemap projection SEI message with `cmp_cancel_flag` equal to 0 that applies to the picture shall not be present. Decoders shall ignore cubemap projection SEI messages when a frame packing arrangement SEI message with `frame_packing_arrangement_cancel_flag` equal to 0 that applies to the picture is present that is not an effectively applicable frame packing arrangement SEI message.

When a segmented rectangular frame packing arrangement SEI message with `segmented_rect_frame_packing_arrangement_cancel_flag` equal to 0 is present that applies to the picture, a cubemap projection SEI message with `cmp_cancel_flag` equal to 0 shall not be present that applies to the picture. Decoders shall ignore cubemap projection SEI messages when a segmented rectangular frame packing arrangement SEI message with `segmented_rect_frame_packing_arrangement_cancel_flag` equal to 0 is present that applies to the picture.

cmp_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous cubemap projection SEI message in output order. `cmp_cancel_flag` equal to 0 indicates that cubemap projection information follows.

cmp_persistence_flag specifies the persistence of the cubemap projection SEI message for the current layer.

`cmp_persistence_flag` equal to 0 specifies that the cubemap projection SEI message applies to the current decoded picture only.

Let `picA` be the current picture. `cmp_persistence_flag` equal to 1 specifies that the cubemap projection SEI message persists for the current layer in output order until one or more of the following conditions are true:

- A new CLVS of the current layer begins.
- The bitstream ends.
- A picture `picB` in the current layer in an access unit containing a cubemap projection SEI message that is applicable to the current layer is output for which `PicOrderCnt(picB)` is greater than `PicOrderCnt(picA)`, where `PicOrderCnt(picB)` and `PicOrderCnt(picA)` are the `PicOrderCntVal` values of `picB` and `picA`, respectively, immediately after the invocation of the decoding process for picture order count for `picB`.

D.3.41.3 Fisheye video information SEI message semantics

The presence of the fisheye video information SEI message for any picture of a CLVS indicates that the picture is a fisheye video picture containing a number of active areas captured by fisheye camera lens. The information carried in the fisheye video information SEI message enables remapping of the colour samples of the pictures onto a sphere coordinate space in sphere coordinates (ϕ , θ), for use in omnidirectional video applications for which the viewing perspective is from the origin looking outward toward the inside of the sphere. The sphere coordinates are defined so that ϕ is the azimuth (longitude, increasing eastward) and θ is the elevation (latitude, increasing northward).

When a fisheye video information SEI message is present for any picture of a CLVS of a particular layer, a fisheye video information SEI message shall be present for the first picture of the CLVS and no equirectangular projection SEI message or cubemap projection SEI message shall be present for any picture of the CLVS.

When `general_non_packed_constraint_flag` is equal to 1 in the active SPS for the current layer, there shall be no fisheye video information SEI messages applicable for any picture of the CLVS of the current layer.

When `aspect_ratio_idc` is present and greater than 1 in the active SPS for the current layer, there should be no fisheye video information SEI messages applicable for any picture of the CLVS of the current layer.

When a frame packing arrangement SEI message with `frame_packing_arrangement_cancel_flag` equal to 0 or a segmented rectangular frame packing arrangement SEI message with `segmented_rect_frame_packing_arrangement_cancel_flag` equal to 0 that applies to the picture is present, a fisheye video information SEI message with `fisheye_cancel_flag` equal to 0 that applies to the picture shall not be present. Decoders shall ignore fisheye video information SEI messages when a frame packing arrangement SEI message with `frame_packing_arrangement_cancel_flag` equal to 0 or a segmented rectangular frame packing arrangement SEI message with `segmented_rect_frame_packing_arrangement_cancel_flag` equal to 0 that applies to the picture is present.

fisheye_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous fisheye video information SEI message in output order. **fisheye_cancel_flag** equal to 0 indicates that fisheye video information follows.

fisheye_persistence_flag specifies the persistence of the fisheye video information SEI message for the current layer.

fisheye_persistence_flag equal to 0 specifies that the fisheye video information SEI message applies to the current decoded picture only.

Let **picA** be the current picture. **fisheye_persistence_flag** equal to 1 specifies that the fisheye video information SEI message persists for the current layer in output order until one or more of the following conditions are true:

- A new CLVS of the current layer begins.
- The bitstream ends.
- A picture **picB** in the current layer in an access unit containing a fisheye video information SEI message that is applicable to the current layer is output for which **PicOrderCnt(picB)** is greater than **PicOrderCnt(picA)**, where **PicOrderCnt(picB)** and **PicOrderCnt(picA)** are the **PicOrderCntVal** values of **picB** and **picA**, respectively, immediately after the invocation of the decoding process for picture order count for **picB**.

fisheye_view_dimension_idc indicates the alignment and viewing direction of a fisheye lens, as follows:

- **fisheye_view_dimension_idc** equal to 0 indicates that **fisheye_num_active_areas** is equal to 2, and the values of **fisheye_camera_centre_azimuth**, **fisheye_camera_centre_elevation**, **fisheye_camera_centre_tilt**, **fisheye_camera_centre_offset_x**, **fisheye_camera_centre_offset_y**, and **fisheye_camera_centre_offset_z** are such that the active areas have aligned optical axes and face opposite directions, and the sum of **fisheye_field_of_view** values is greater than or equal to $360 * 2^{16}$.
- **fisheye_view_dimension_idc** equal to 1 indicates that **fisheye_num_active_areas** is equal to 2, and the values of **fisheye_camera_centre_azimuth**, **fisheye_camera_centre_elevation**, **fisheye_camera_centre_tilt**, **fisheye_camera_centre_offset_x**, **fisheye_camera_centre_offset_y**, and **fisheye_camera_centre_offset_z** are such that the active areas have parallel optical axes that are orthogonal to the line intersecting the camera centre points, and the camera corresponding to **i** equal to 0 is the left view.
- **fisheye_view_dimension_idc** equal to 2 indicates that **fisheye_num_active_areas** is equal to 2, and the values of **fisheye_camera_centre_azimuth**, **fisheye_camera_centre_elevation**, **fisheye_camera_centre_tilt**, **fisheye_camera_centre_offset_x**, **fisheye_camera_centre_offset_y**, and **fisheye_camera_centre_offset_z** are such that the active areas have parallel optical axes that are orthogonal to the line intersecting the camera centre points, and the camera corresponding to **i** equal to 0 is the right view.
- **fisheye_view_dimension_idc** equal to 7 indicates that no additional constraints are implied for the syntax element values within the fisheye video information SEI message.
- Values of **fisheye_view_dimension_idc** in the range of 3 to 6, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of **fisheye_view_dimension_idc** in the range of 3 to 6, inclusive, shall ignore it.

fisheye_reserved_zero_3bits shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for **fisheye_reserved_zero_3bits** are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of **fisheye_reserved_zero_3bits**.

fisheye_num_active_areas_minus1 plus 1 specifies the number of active areas in the coded picture. The value of **fisheye_num_active_areas_minus1** shall be in the range of 0 to 3, inclusive. Values of **fisheye_num_active_areas_minus1** greater than 3 are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a fisheye video information SEI message with **fisheye_num_active_areas_minus1** greater than 3 shall ignore the fisheye video information SEI message.

fisheye_circular_region_centre_x[i] and **fisheye_circular_region_centre_y[i]** specify the horizontal and vertical coordinates of the centre of the circular region that contains the **i**-th active area in the coded picture, respectively, in units of 2^{-16} luma samples. The value of **fisheye_circular_region_centre_x[i]** and **fisheye_circular_region_centre_y[i]** shall be in the range of 0 to $65\,536 * 2^{16} - 1$ (i.e., 4 294 967 295), inclusive.

fisheye_rect_region_top[i], **fisheye_rect_region_left[i]**, **fisheye_rect_region_width[i]**, and **fisheye_rect_region_height[i]** specify the coordinates of the top-left corner and the width and height of the **i**-th rectangular region that contains the **i**-th active area, in units of luma samples.

The value of **fisheye_rect_region_top[i]** shall be in the range of **SubHeightC * conf_win_top_offset** to **pic_height_in_luma_samples - (SubHeightC * conf_win_bottom_offset + 1)**, inclusive.

The value of **fisheye_rect_region_left[i]** shall be in the range of **SubWidthC * conf_win_left_offset** to **pic_width_in_luma_samples - (SubWidthC * conf_win_right_offset + 1)**, inclusive.

The value of `fisheye_rect_region_width[i]` shall be in the range of 1 to $\text{pic_width_in_luma_samples} - \text{SubWidthC} * (\text{conf_win_left_offset} + \text{conf_win_right_offset})$, inclusive.

The value of `fisheye_rect_region_height[i]` shall be in the range of 1 to $\text{pic_height_in_luma_samples} - \text{SubHeightC} * (\text{conf_win_top_offset} + \text{conf_win_bottom_offset})$, inclusive.

The sum of `fisheye_rect_region_top[i]` and `fisheye_rect_region_height[i]` shall be less than or equal to $\text{pic_height_in_luma_samples} - \text{SubHeightC} * \text{conf_win_bottom_offset}$.

The sum of `fisheye_rect_region_left[i]` and `fisheye_rect_region_width[i]` shall be less than or equal to $\text{pic_width_in_luma_samples} - \text{SubWidthC} * \text{conf_win_right_offset}$.

`fisheye_circular_region_radius[i]` specifies the radius of the circular region that contains the *i*-th active area that is defined as a length from the centre of the circular region specified by `fisheye_circular_region_centre_x[i]` and `fisheye_circular_region_centre_y[i]` to the outermost pixel boundary of the circular region, in units of 2^{-16} luma samples, that corresponds to the maximum field of view of the *i*-th fisheye lens, specified by `fisheye_field_of_view[i]`. The value of `fisheye_circular_region_radius[i]` shall be in the range of 0 to $65\,536 * 2^{16} - 1$ (i.e., 4 294 967 295), inclusive.

The *i*-th active area is defined as the intersection of the *i*-th rectangular region, specified by `fisheye_rect_region_top[i]`, `fisheye_rect_region_left[i]`, `fisheye_rect_region_width[i]`, and `fisheye_rect_region_height[i]`, and the *i*-th circular region, specified by `fisheye_circular_region_centre_x[i]`, `fisheye_circular_region_centre_y[i]`, and `fisheye_circular_region_radius[i]`.

Each active area shall contain at least one sample location. There shall not be any sample location that is within more than one active area.

`fisheye_scene_radius[i]` specifies the radius of a circular region within the *i*-th active area in units of 2^{-16} luma samples, where the obstruction, such as the camera body, is not included in the region specified by `fisheye_circular_region_centre_x[i]`, `fisheye_circular_region_centre_y[i]`, and `fisheye_scene_radius[i]`. The value of `fisheye_scene_radius[i]` shall be less than or equal to `fisheye_circular_region_radius[i]`, and shall be in the range of 0 to $65\,536 * 2^{16} - 1$ (i.e., 4 294 967 295), inclusive. The enclosed area is the suggested area for stitching as recommended by the encoder.

`fisheye_camera_centre_azimuth[i]` and **`fisheye_camera_centre_elevation[i]`** indicate the sphere coordinates that correspond to the centre of the circular region that contains the *i*-th active area in the cropped output picture, in units of 2^{-16} degrees. The value of `fisheye_camera_centre_azimuth[i]` shall be in the range of $-180 * 2^{16}$ (i.e., -11 796 480) to $180 * 2^{16} - 1$ (i.e., 11 796 479), inclusive, and the value of `fisheye_camera_centre_elevation[i]` shall be in the range of $-90 * 2^{16}$ (i.e., -5 898 240) to $90 * 2^{16}$ (i.e., 5 898 240), inclusive.

`fisheye_camera_centre_tilt[i]` indicates the tilt angle of the sphere region that corresponds to the *i*-th active area of the cropped output picture, in units of 2^{-16} degrees. The value of `fisheye_camera_centre_tilt[i]` shall be in the range of $-180 * 2^{16}$ (i.e., -11 796 480) to $180 * 2^{16} - 1$ (i.e., 11 796 479), inclusive.

`fisheye_camera_centre_offset_x[i]`, **`fisheye_camera_centre_offset_y[i]`** and **`fisheye_camera_centre_offset_z[i]`** indicate the XYZ offset values, in units of 2^{-16} millimeters, of the focal centre of the fisheye camera lens corresponding to the *i*-th active area from the focal centre origin of the overall fisheye camera configuration. The value of each of `fisheye_camera_centre_offset_x[i]`, `fisheye_camera_centre_offset_y[i]`, and `fisheye_camera_centre_offset_z[i]` shall be in the range of 0 to $65\,536 * 2^{16} - 1$ (i.e., 4 294 967 295), inclusive.

`fisheye_field_of_view[i]` specifies the field of view of the lens that corresponds to the *i*-th active area in the coded picture, in units of 2^{-16} degrees. The value of `fisheye_field_of_view[i]` shall be in the range of 0 to $360 * 2^{16}$ (i.e., 23 592 960), inclusive.

`fisheye_num_polynomial_coeffs[i]` specifies the number of polynomial coefficients for the circular region corresponding to the *i*-th active area. The value of `fisheye_num_polynomial_coeffs[i]` shall be in the range of 0 to 8, inclusive. Values of `fisheye_num_polynomial_coeffs[i]` greater than 8 are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a fisheye video information SEI message with `fisheye_num_polynomial_coeffs[i]` greater than 8 shall ignore the fisheye video information SEI message.

`fisheye_polynomial_coeff[i][j]` specifies the *j*-th polynomial coefficient value, in units of 2^{-24} , of the curve function that maps the normalized distance of a luma sample from the centre of the circular region corresponding to the *i*-th active area to the angular value of a sphere coordinate from the normal vector of a nominal imaging plane that passes through the centre of the sphere coordinate system for the *i*-th active region. The value of `fisheye_polynomial_coeff[i][j]` shall be in the range of $-128 * 2^{24}$ (i.e., -2 147 483 648) to $128 * 2^{24} - 1$ (i.e., 2 147 483 647), inclusive.

D.3.41.4 Sphere rotation SEI message semantics

The sphere rotation SEI message provides information on rotation angles yaw (α), pitch (β), and roll (γ) that are used for conversion between the global coordinate axes and the local coordinate axes.

Relative to an (x, y, z) Cartesian coordinate system, yaw expresses a rotation around the z (vertical, up) axis, pitch rotates around the y (lateral, side-to-side) axis, and roll rotates around the x (back-to-front) axis. Rotations are extrinsic, i.e., around x, y, and z fixed reference axes. The angles increase clockwise when looking from the origin towards the positive end of an axis.

sphere_rotation_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous sphere rotation SEI message in output order. sphere_rotation_cancel_flag equal to 0 indicates that sphere rotation information follows.

sphere_rotation_persistence_flag specifies the persistence of the sphere rotation SEI message for the current layer.

sphere_rotation_persistence_flag equal to 0 specifies that the sphere rotation SEI message applies to the current decoded picture only.

Let picA be the current picture. sphere_rotation_persistence_flag equal to 1 specifies that the sphere rotation SEI message persists for the current layer in output order until one or more of the following conditions are true:

- A new CLVS of the current layer begins.
- The bitstream ends.
- A picture picB in the current layer in an access unit containing a sphere rotation SEI message that is applicable to the current layer is output for which $\text{PicOrderCnt}(\text{picB})$ is greater than $\text{PicOrderCnt}(\text{picA})$, where $\text{PicOrderCnt}(\text{picB})$ and $\text{PicOrderCnt}(\text{picA})$ are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

When an equirectangular projection SEI message with **erp_cancel_flag** equal to 0 or a cubemap projection SEI message with **cmp_cancel_flag** equal to 0 is not present in the CLVS that applies to the current picture and precedes the sphere rotation SEI message in decoding order, a sphere rotation SEI message with **sphere_rotation_cancel_flag** equal to 0 shall not be present in the CLVS that applies to the current picture. Decoders shall ignore sphere rotation SEI messages with **sphere_rotation_cancel_flag** equal to 0 that do not follow, in decoding order, an equirectangular projection SEI message with **erp_cancel_flag** equal to 0 or a cubemap projection SEI message with **cmp_cancel_flag** equal to 0 in the CLVS that applies to the current picture.

sphere_rotation_reserved_zero_6bits shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for sphere_rotation_reserved_zero_6bits are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of sphere_rotation_reserved_zero_6bits.

yaw_rotation specifies the value of the yaw rotation angle, in units of 2^{-16} degrees. The value of yaw_rotation shall be in the range of $-180 * 2^{16}$ (i.e., -11 796 480) to $180 * 2^{16} - 1$ (i.e., 11 796 479), inclusive. When not present, the value of yaw_rotation is inferred to be equal to 0.

pitch_rotation specifies the value of the pitch rotation angle, in units of 2^{-16} degrees. The value of pitch_rotation shall be in the range of $-90 * 2^{16}$ (i.e., -5 898 240) to $90 * 2^{16}$ (i.e., 5 898 240), inclusive. When not present, the value of pitch_rotation is inferred to be equal to 0.

roll_rotation specifies the value of the roll rotation angle, in units of 2^{-16} degrees. The value of roll_rotation shall be in the range of $-180 * 2^{16}$ (i.e., -11 796 480) to $180 * 2^{16} - 1$ (i.e., 11 796 479), inclusive. When not present, the value of roll_rotation is inferred to be equal to 0.

D.3.41.5 Region-wise packing SEI message semantics

The region-wise packing SEI message provides information to enable remapping of the colour samples of the cropped decoded pictures onto projected pictures as well as information on the location and size of the guard bands, if any.

rwp_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous region-wise packing SEI message in output order. rwp_cancel_flag equal to 0 indicates that region-wise packing information follows.

rwp_persistence_flag specifies the persistence of the region-wise packing SEI message for the current layer.

rwp_persistence_flag equal to 0 specifies that the region-wise packing SEI message applies to the current decoded picture only.

Let picA be the current picture. rwp_persistence_flag equal to 1 specifies that the region-wise packing SEI message persists for the current layer in output order until one or more of the following conditions are true:

- A new CLVS of the current layer begins.
- The bitstream ends.
- A picture picB in the current layer in an access unit containing a region-wise packing SEI message that is applicable to the current layer is output for which $\text{PicOrderCnt}(\text{picB})$ is greater than $\text{PicOrderCnt}(\text{picA})$, where

PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

When an equirectangular projection SEI message with `erp_cancel_flag` equal to 0 and `erp_guard_band_flag` equal to 0 or a cubemap projection SEI message with `cmp_cancel_flag` equal to 0 is not present in the CLVS that applies to the current picture and precedes the region-wise packing SEI message in decoding order, a region-wise packing SEI message with `rwp_cancel_flag` equal to 0 shall not be present in the CLVS that applies to the current picture. Decoders shall ignore region-wise packing SEI messages with `rwp_cancel_flag` equal to 0 that do not follow, in decoding order, an equirectangular projection SEI message with `erp_cancel_flag` equal to 0 or a cubemap projection SEI message with `cmp_cancel_flag` equal to 0 in the CLVS that applies to the current picture.

For the frame packing arrangement scheme indicated by a frame packing arrangement SEI message that applies to the current picture, if a region-wise packing SEI message with `rwp_cancel_flag` equal to 0 is present that applies to the current picture, the frame packing arrangement scheme applies to the projected picture, otherwise, the frame packing arrangement scheme applies to the cropped decoded picture.

If a frame packing arrangement SEI message with `frame_packing_arrangement_cancel_flag` equal to 0, `frame_packing_arrangement_type` equal to 3, 4, or 5, and `quincunx_sampling_flag` equal to 0 is not present that applies to the current picture, the variables `StereoFlag`, `TopBottomFlag`, `SideBySideFlag`, and `TempInterleavingFlag` are all set equal to 0, the variables `HorDiv1` and `VerDiv1` are both set equal to 1. Otherwise the following applies:

- `StereoFlag` is set equal to 1.
- When the `frame_packing_arrangement_type` is equal to 3, `SideBySideFlag` is set equal to 1, `TopBottomFlag` and `TempInterleavingFlag` are both set equal to 0, `HorDiv1` is set equal to 2 and `VerDiv1` is set equal to 1.
- When the `frame_packing_arrangement_type` is equal to 4, `TopBottomFlag` is set equal to 1, `SideBySideFlag` and `TempInterleavingFlag` are both set equal to 0, `HorDiv1` is set equal to 1 and `VerDiv1` is set equal to 2.
- When the `frame_packing_arrangement_type` is equal to 5, `TempInterleavingFlag` is set equal to 1, `TopBottomFlag` and `SideBySideFlag` are both set equal to 0, `HorDiv1` and `VerDiv1` are both set equal to 1.

constituent_picture_matching_flag equal to 1 specifies that the projected region information, packed region information, and guard band region information in this SEI message apply individually to each constituent picture and that the packed picture and the projected picture have the same stereoscopic frame packing format indicated by the frame packing arrangement SEI message. **constituent_picture_matching_flag** equal to 0 specifies that the projected region information, packed region information, and guard band region information in this SEI message apply to the projected picture.

When either of the following conditions is true, the value of **constituent_picture_matching_flag** shall be equal to 0:

- `StereoFlag` is equal to 0.
- `StereoFlag` is equal to 1 and `frame_packing_arrangement_type` is equal to 5.

rwp_reserved_zero_5bits shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for `rwp_reserved_zero_5bits[i]` are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of `rwp_reserved_zero_5bits[i]`.

num_packed_regions specifies the number of packed regions when **constituent_picture_matching_flag** is equal to 0. The value of **num_packed_regions** shall be greater than 0. When **constituent_picture_matching_flag** is equal to 1, the total number of packed regions is equal to **num_packed_regions** * 2, and the information in each entry of the loop of **num_packed_regions** entries applies to each constituent picture of the projected picture and the packed picture.

proj_picture_width and **proj_picture_height** specify the width and height, respectively, of the projected picture, in relative projected picture sample units.

The values of **proj_picture_width** and **proj_picture_height** shall both be greater than 0.

packed_picture_width and **packed_picture_height** specify the width and height, respectively, of the packed picture, in relative packed picture sample units.

The values of **packed_picture_width** and **packed_picture_height** shall both be greater than 0.

It is a requirement of bitstream conformance that **packed_picture_width** and **packed_picture_height** shall have such values that **packed_picture_width** is an integer multiple of **cropPicWidth** and **packed_picture_height** is an integer multiple of **cropPicHeight**, where **cropPicWidth** and **cropPicHeight** are the width and height, respectively, of the cropped decoded picture.

rwp_reserved_zero_4bits[i] shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for `rwp_reserved_zero_4bits[i]` are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of `rwp_reserved_zero_4bits[i]`.

rwpx_transform_type[i] specifies the rotation and mirroring to be applied to the i-th packed region to remap to the i-th projected region. When **rwpx_transform_type[i]** specifies both rotation and mirroring, rotation applies before mirroring. The values of **rwpx_transform_type[i]** are specified in Table D.22.

Table D.22 – rwpx_transform_type[i] values

Value	Description
0	no transform
1	mirroring horizontally
2	rotation by 180 degrees (anticlockwise)
3	rotation by 180 degrees (anticlockwise) before mirroring horizontally
4	rotation by 90 degrees (anticlockwise) before mirroring horizontally
5	rotation by 90 degrees (anticlockwise)
6	rotation by 270 degrees (anticlockwise) before mirroring horizontally
7	rotation by 270 degrees (anticlockwise)

rwpx_guard_band_flag[i] equal to 0 specifies that the i-th packed region does not have a guard band. **rwpx_guard_band_flag[i]** equal to 1 specifies that the i-th packed region has a guard band.

proj_region_width[i], **proj_region_height[i]**, **proj_region_top[i]** and **proj_region_left[i]** specify the width, height, top sample row, and the left-most sample column, respectively, of the i-th projected region, either within the projected picture (when **constituent_picture_matching_flag** is equal to 0) or within the constituent picture of the projected picture (when **constituent_picture_matching_flag** is equal to 1).

proj_region_width[i], **proj_region_height[i]**, **proj_region_top[i]**, and **proj_region_left[i]** are indicated in relative projected picture sample units.

NOTE 1 – Two projected regions may partially or entirely overlap with each other.

packed_region_width[i], **packed_region_height[i]**, **packed_region_top[i]**, and **packed_region_left[i]** specify the width, height, the top luma sample row, and the left-most luma sample column, respectively, of the packed region, either within the region-wise packed picture (when **constituent_picture_matching_flag** is equal to 0) or within each constituent picture of the region-wise packed picture (when **constituent_picture_matching_flag** is equal to 1).

packed_region_width[i], **packed_region_height[i]**, **packed_region_top[i]**, and **packed_region_left[i]** are indicated in relative region-wise packed picture sample units. **packed_region_width[i]**, **packed_region_height[i]**, **packed_region_top[i]**, and **packed_region_left[i]** shall represent integer horizontal and vertical coordinates of luma sample units within the cropped decoded pictures.

NOTE 2 – Two packed regions may partially or entirely overlap with each other.

rwpx_left_guard_band_width[i] specifies the width of the guard band on the left side of the i-th packed region in relative region-wise packed picture sample units. When **chroma_format_idc** is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format), **rwpx_left_guard_band_width[i]** shall correspond to an even number of luma samples within the cropped decoded picture.

rwpx_right_guard_band_width[i] specifies the width of the guard band on the right side of the i-th packed region in relative region-wise packed picture sample units. When **chroma_format_idc** is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format), **rwpx_right_guard_band_width[i]** shall correspond to an even number of luma samples within the cropped decoded picture.

rwpx_top_guard_band_height[i] specifies the height of the guard band above the i-th packed region in relative region-wise packed picture sample units. When **chroma_format_idc** is equal to 1 (4:2:0 chroma format), **rwpx_top_guard_band_height[i]** shall correspond to an even number of luma samples within the cropped decoded picture.

rwpx_bottom_guard_band_height[i] specifies the height of the guard band below the i-th packed region in relative region-wise packed picture sample units. When **chroma_format_idc** is equal to 1 (4:2:0 chroma format), **rwpx_bottom_guard_band_height[i]** shall correspond to an even number of luma samples within the cropped decoded picture.

When **rwpx_guard_band_flag[i]** is equal to 1, **rwpx_left_guard_band_width[i]**, **rwpx_right_guard_band_width[i]**, **rwpx_top_guard_band_height[i]**, or **rwpx_bottom_guard_band_height[i]** shall be greater than 0.

The i-th packed region as specified by this SEI message shall not overlap with any other packed region specified by the same SEI message or any guard band specified by the same SEI message.

The guard bands associated with the i -th packed region, if any, as specified by this SEI message shall not overlap with any packed region specified by the same SEI message or any other guard bands specified by the same SEI message.

rwg_guard_band_not_used_for_pred_flag[i] equal to 0 specifies that the guard bands may or may not be used in the inter prediction process. **rwg_guard_band_not_used_for_pred_flag[i]** equal to 1 specifies that the sample values of the guard bands are not used in the inter prediction process.

NOTE 3 – When **rwg_guard_band_not_used_for_pred_flag[i]** is equal to 1, the sample values within guard bands in cropped decoded pictures can be rewritten even if the cropped decoded pictures were used as references for inter prediction of subsequent pictures to be decoded. For example, the content of a packed region can be seamlessly expanded to its guard band with decoded and re-projected samples of another packed region.

rwg_guard_band_type[i][j] indicates the type of the guard bands for the i -th packed region as follows, with j equal to 0, 1, 2, or 3 indicating that the semantics below apply to the left, right, top, or bottom edge, respectively, of the packed region:

- **rwg_guard_band_type[i][j]** equal to 0 indicates that the content of the guard bands in relation to the content of the packed regions is unspecified. When **rwg_guard_band_not_used_for_pred_flag[i]** is equal to 0, **rwg_guard_band_type[i][j]** shall not be equal to 0.
- **rwg_guard_band_type[i][j]** equal to 1 indicates that the content of the guard bands suffices for interpolation of sample values at sub-pel sample fractional locations within the packed region and less than one sample outside of the boundary of the packed region.

NOTE 4 – **rwg_guard_band_type[i][j]** equal to 1 can be used when the boundary samples of a packed region have been copied horizontally or vertically to the guard band.

- **rwg_guard_band_type[i][j]** equal to 2 indicates that the content of the guard bands represents actual picture content that is spherically adjacent to the content in the packed region and is on the surface of the packed region at quality that gradually changes from the picture quality of the packed region to that of the spherically adjacent packed region.
- **rwg_guard_band_type[i][j]** equal to 3 indicates that the content of the guard bands represents actual picture content that is spherically adjacent to the content in the packed region and is on the surface of the packed region at a similar picture quality as within the packed region.
- **rwg_guard_band_type[i][j]** values greater than 3 are reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value of **rwg_guard_band_type[i][j]** when the value is greater than 3 as equivalent to the value 0.

rwg_guard_band_reserved_zero_3bits[i] shall be equal to 0 in bitstreams conforming to this version of this Specification. Other values for **rwg_guard_band_reserved_zero_3bits[i]** are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of **rwg_guard_band_reserved_zero_3bits[i]**.

The variables **NumPackedRegions**, **PackedRegionLeft[n]**, **PackedRegionTop[n]**, **PackedRegionWidth[n]**, **PackedRegionHeight[n]**, **ProjRegionLeft[n]**, **ProjRegionTop[n]**, **ProjRegionWidth[n]**, **ProjRegionHeight[n]**, and **TransformType[n]** are derived as follows:

- For n in the range of 0 to **num_packed_regions** – 1, inclusive, the following applies:
 - **PackedRegionLeft[n]** is set equal to **packed_region_left[n]**.
 - **PackedRegionTop[n]** is set equal to **packed_region_top[n]**.
 - **PackedRegionWidth[n]** is set equal to **packed_region_width[n]**.
 - **PackedRegionHeight[n]** is set equal to **packed_region_height[n]**.
 - **ProjRegionLeft[n]** is set equal to **proj_region_left[n]**.
 - **ProjRegionTop[n]** is set equal to **proj_region_top[n]**.
 - **ProjRegionWidth[n]** is set equal to **proj_region_width[n]**.
 - **ProjRegionHeight[n]** is set equal to **proj_region_height[n]**.
 - **TransformType[n]** is set equal to **rwg_transform_type[n]**.
- If **constituent_picture_matching_flag** is equal to 0, the following applies:
 - **NumPackedRegions** is set equal to **num_packed_regions**.
- Otherwise (**constituent_picture_matching_flag** is equal to 1), the following applies:
 - **NumPackedRegions** is set equal to $2 * \text{num_packed_regions}$.

- When TopBottomFlag is equal to 1, the following applies:
 - projLeftOffset and packedLeftOffset are both set equal to 0.
 - projTopOffset is set equal to $\text{proj_picture_height} / 2$ and packedTopOffset is set equal to $\text{packed_picture_height} / 2$.
- When SideBySideFlag is equal to 1, the following applies:
 - projLeftOffset is set equal to $\text{proj_picture_width} / 2$ and packedLeftOffset is set equal to $\text{packed_picture_width} / 2$.
 - projTopOffset and packedTopOffset are both set equal to 0.
- For n in the range of $\text{NumPackedRegions} / 2$ to $\text{NumPackedRegions} - 1$, inclusive, the following applies:
 - $n\text{Idx}$ is set equal to $n - \text{NumPackedRegions} / 2$.
 - $\text{PackedRegionLeft}[n]$ is set equal to $\text{packed_region_left}[n\text{Idx}] + \text{packedLeftOffset}$.
 - $\text{PackedRegionTop}[n]$ is set equal to $\text{packed_region_top}[n\text{Idx}] + \text{packedTopOffset}$.
 - $\text{PackedRegionWidth}[n]$ is set equal to $\text{packed_region_width}[n\text{Idx}]$.
 - $\text{PackedRegionHeight}[n]$ is set equal to $\text{packed_region_height}[n\text{Idx}]$.
 - $\text{ProjRegionLeft}[n]$ is set equal to $\text{proj_region_left}[n\text{Idx}] + \text{projLeftOffset}$.
 - $\text{ProjRegionTop}[n]$ is set equal to $\text{proj_region_top}[n\text{Idx}] + \text{projTopOffset}$.
 - $\text{ProjRegionWidth}[n]$ is set equal to $\text{proj_region_width}[n\text{Idx}]$.
 - $\text{ProjRegionHeight}[n]$ is set equal to $\text{proj_region_height}[n\text{Idx}]$.
 - $\text{TransformType}[n]$ is set equal to $\text{rwp_transform_type}[n\text{Idx}]$.

For each value of n in the range of 0 to $\text{NumPackedRegions} - 1$, inclusive, the values of $\text{ProjRegionWidth}[n]$, $\text{ProjRegionHeight}[n]$, $\text{ProjRegionTop}[n]$, and $\text{ProjRegionLeft}[n]$ are constrained as follows:

- $\text{ProjRegionWidth}[n]$ shall be in the range of 1 to $\text{proj_picture_width}$, inclusive.
- $\text{ProjRegionHeight}[n]$ shall be in the range of 1 to $\text{proj_picture_height}$, inclusive.
- $\text{ProjRegionLeft}[n]$ shall be in the range of 0 to $\text{proj_picture_width} - 1$, inclusive.
- $\text{ProjRegionTop}[n]$ shall be in the range of 0 to $\text{proj_picture_height} - 1$, inclusive.
- If $\text{ProjRegionTop}[n]$ is less than $\text{proj_picture_height} / \text{VerDiv1}$, the sum of $\text{ProjRegionTop}[n]$ and $\text{ProjRegionHeight}[n]$ shall be less than or equal to $\text{proj_picture_height} / \text{VerDiv1}$. Otherwise, the sum of $\text{ProjRegionTop}[n]$ and $\text{ProjRegionHeight}[n]$ shall be less than or equal to $\text{proj_picture_height} / \text{VerDiv1} * 2$.

For each value of n in the range of 0 to $\text{NumPackedRegions} - 1$, inclusive, the values of $\text{PackedRegionWidth}[n]$, $\text{PackedRegionHeight}[n]$, $\text{PackedRegionTop}[n]$, and $\text{PackedRegionLeft}[n]$ are constrained as follows:

- $\text{PackedRegionWidth}[n]$ shall be in the range of 1 to $\text{packed_picture_width}$, inclusive.
- $\text{PackedRegionHeight}[n]$ shall be in the range of 1 to $\text{packed_picture_height}$, inclusive.
- $\text{PackedRegionLeft}[n]$ shall be in the range of 0 to $\text{packed_picture_width} - 1$, inclusive.
- $\text{PackedRegionTop}[n]$ shall be in the range of 0 to $\text{packed_picture_height} - 1$, inclusive.
- If $\text{PackedRegionLeft}[n]$ is less than $\text{packed_picture_width} / \text{HorDiv1}$, the sum of $\text{PackedRegionLeft}[n]$ and $\text{PackedRegionWidth}[n]$ shall be less than or equal to $\text{packed_picture_width} / \text{HorDiv1}$. Otherwise, the sum of $\text{PackedRegionLeft}[n]$ and $\text{PackedRegionWidth}[n]$ shall be less than or equal to $\text{packed_picture_width} / \text{HorDiv1} * 2$.
- If $\text{PackedRegionTop}[n]$ is less than $\text{packed_picture_height} / \text{VerDiv1}$, the sum of $\text{PackedRegionTop}[n]$ and $\text{PackedRegionHeight}[n]$ shall be less than or equal to $\text{packed_picture_height} / \text{VerDiv1}$. Otherwise, the sum of $\text{PackedRegionTop}[n]$ and $\text{PackedRegionHeight}[n]$ shall be less than or equal to $\text{packed_picture_height} / \text{VerDiv1} * 2$.
- When chroma_format_idc is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format), $\text{PackedRegionLeft}[n]$ shall correspond to an even horizontal coordinate value of luma sample units, and $\text{PackedRegionWidth}[n]$ shall correspond to an even number of luma samples, both within the decoded picture.
- When chroma_format_idc is equal to 1 (4:2:0 chroma format), $\text{PackedRegionTop}[n]$ shall correspond to an even vertical coordinate value of luma sample units, and $\text{ProjRegionHeight}[n]$ shall correspond to an even number of luma samples, both within the decoded picture.

D.3.41.6 Omnidirectional viewport SEI message semantics

The omnidirectional viewport SEI message specifies the coordinates of one or more regions of sphere-coordinate geometry, bounded by four great circles, corresponding to viewports recommended for display when the user does not have control of the viewing orientation or has released control of the viewing orientation.

When an effectively applicable frame packing arrangement SEI message, as specified in clause D.3.41.1 or D.3.41.2, that applies to the picture is present, the information indicated by the omnidirectional viewport SEI message applies to both views.

omni_viewport_id contains an identifying number that may be used to identify the purpose of the one or more recommended viewport regions.

omni_viewport_id equal to 0 indicates that the recommended viewports are per "director's cut", i.e., a viewport suggested according to the creative intent of the content author or content provider. omni_viewport_id equal to 1 indicates that the recommended viewports are selected based on measurements of viewing statistics.

Values of omni_viewport_id from 2 to 511, inclusive, may be used as determined by the application. Values of omni_viewport_id from 512 to 1 023 are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of omni_viewport_id in the range of 512 to 1 023, inclusive, shall ignore it.

omni_viewport_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous omnidirectional viewport SEI message in output order. omni_viewport_cancel_flag equal to 0 indicates that omnidirectional viewport information follows.

omni_viewport_persistence_flag specifies the persistence of the omnidirectional viewport SEI message for the current layer.

omni_viewport_persistence_flag equal to 0 specifies that the omnidirectional viewport SEI message applies to the current decoded picture only.

Let picA be the current picture. omni_viewport_persistence_flag equal to 1 specifies that the omnidirectional viewport SEI message persists for the current layer in output order until one or more of the following conditions are true:

- A new CLVS of the current layer begins.
- The bitstream ends.
- A picture picB in the current layer in an access unit containing an omnidirectional viewport SEI message that is applicable to the current layer is output for which PicOrderCnt(picB) is greater than PicOrderCnt(picA), where PicOrderCnt(picB) and PicOrderCnt(picA) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

When an equirectangular projection SEI message with erp_cancel_flag equal to 0 or a cubemap projection SEI message with cmp_cancel_flag equal to 0 is not present in the CLVS that applies to the current picture and precedes the omnidirectional viewport SEI message in decoding order, an omnidirectional viewport SEI message with omni_viewport_cancel_flag equal to 0 shall not be present in the CLVS that applies to the current picture. Decoders shall ignore omnidirectional viewport SEI messages with omni_viewport_cancel_flag equal to 0 that do not follow, in decoding order, an equirectangular projection SEI message with erp_cancel_flag equal to 0 or a cubemap projection SEI message with cmp_cancel_flag equal to 0 in the CLVS that applies to the current picture.

omni_viewport_cnt_minus1 plus 1 specifies the number of recommended viewport regions that are indicated by the SEI message.

When omni_viewport_cnt_minus1 is greater than 0 and there is no information provided by external means not specified in this Specification on which recommended viewport is suggested to be displayed, the following applies:

- When omni_viewport_id is equal to 0 or 1, the 0-th recommended viewport is suggested to be displayed when the user does not have control of the viewing orientation or has released control of the viewing orientation.
- When omni_viewport_id is equal to 0, between any two recommended viewports per director's cut, the i-th recommended viewport has higher priority than the j-th recommended viewport for any values of i and j when i is less than j. The 0-th recommended viewport per director's cut has the highest priority.
- When omni_viewport_id is equal to 1, between any two recommended viewports, the i-th recommended viewport has higher popularity, among some selection of candidate viewports, than the j-th recommended viewport for any values of i and j when i is less than j. The 0-th most-viewed recommended viewport has the highest popularity. The selection of the candidate viewports is outside the scope of this Specification.

omni_viewport_azimuth_centre[i] and **omni_viewport_elevation_centre[i]** indicate the centre of the i-th recommended viewport region, in units of 2^{-16} degrees relative to the global coordinate axes. The value of

`omni_viewport_azimuth_centre[i]` shall be in the range of $-180 * 2^{16}$ (i.e., -11 796 480) to $180 * 2^{16} - 1$ (i.e., 11 796 479), inclusive. The value of `omni_viewport_elevation_centre[i]` shall be in the range of $-90 * 2^{16}$ (i.e., -5 898 240) to $90 * 2^{16}$ (i.e., 5 898 240), inclusive.

`omni_viewport_tilt_centre[i]` indicates the tilt angle of the *i*-th recommended viewport region, in units of 2^{-16} degrees. The value of `omni_viewport_tilt_centre[i]` shall be in the range of $-180 * 2^{16}$ (i.e., -11 796 480) to $2^{16} - 1$ (i.e., 11 796 479), inclusive.

`omni_viewport_hor_range[i]` indicates the azimuth range of the *i*-th recommended viewport region, in units of 2^{-16} degrees. The value of `omni_viewport_hor_range[i]` shall be in the range of 1 to $360 * 2^{16}$ (i.e., 23 592 960), inclusive.

`omni_viewport_ver_range[i]` indicates the elevation range of the *i*-th recommended viewport region, in units of 2^{-16} degrees. The value of `omni_viewport_ver_range[i]` shall be in the range of 1 to $180 * 2^{16}$ (i.e., 11 796 480), inclusive.

D.3.41.7 Sample location remapping process

D.3.41.7.1 General

To remap colour sample locations of a region-wise packed picture to a unit sphere, the following ordered steps are applied:

1. A region-wise packed picture is obtained as the cropped decoded picture by decoding a coded picture. For purposes of interpretation of chroma samples, the input to the indicated remapping process is the set of decoded sample values after applying an (unspecified) upsampling conversion process to the 4:4:4 colour sampling format as necessary when `chroma_format_idc` is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format). This (unspecified) upsampling process should account for the relative positioning relationship between the luma and chroma samples as indicated by `chroma_sample_loc_type_top_field` and `chroma_sample_loc_type_bottom_field`, when present.
2. If region-wise packing is indicated, the sample locations of the region-wise packed picture are converted to sample locations of the respective projected picture as specified in clause D.3.41.7.4. Otherwise, the projected picture is identical to the region-wise packed picture.
3. If frame packing is indicated, the sample locations of the projected picture are converted to sample locations of the respective constituent picture of the projected picture, as specified in clause D.3.41.7.5. Otherwise, the constituent picture of the projected picture is identical to the projected picture.
4. The sample locations of a constituent picture of the projected picture are converted to sphere coordinates relative to the local coordinate axes, as specified in clause D.3.41.7.2.
5. If rotation is indicated, the sphere coordinates relative to the local coordinate axes are converted to sphere coordinates relative to the global coordinate axes, as specified in clause D.3.41.7.3. Otherwise, the global coordinate axes are identical to the local coordinate axes.

The overall process for mapping of luma sample locations within a region-wise packed picture to sphere coordinates relative to the global coordinate axes is normatively specified in clause D.3.41.7.5.

For each region-wise packed picture corresponding to a decoded picture, the following applies:

- When an equirectangular projection SEI message with `erp_cancel_flag` equal to 0 that applies to the picture is present, `ErpFlag` is set equal to 1, and `CmpFlag` is set equal to 0.
- When a cubemap projection SEI message with `cmp_cancel_flag` equal to 0 that applies to the picture is present, `CmpFlag` is set equal to 1, and `ErpFlag` is set equal to 0.
- If a sphere rotation SEI message with `sphere_rotation_cancel_flag` equal to 0 that applies to the picture is present, `RotationFlag` is set equal to 1, and `RotationYaw`, `RotationPitch`, and `RotationRoll` are set equal to `yaw_rotation` $\div 2^{16}$, `pitch_rotation` $\div 2^{16}$, and `roll_rotation` $\div 2^{16}$, respectively.
- Otherwise, `RotationFlag` is set equal to 0.
- If a frame packing arrangement SEI message with `frame_packing_arrangement_cancel_flag` equal to 0 that applies to the picture is not present, `StereoFlag`, `TopBottomFlag`, and `SideBySideFlag` are all set equal to 0, `HorDiv1` is set equal to 1, and `VerDiv1` is set equal to 1.
- Otherwise, the following applies:
 - `StereoFlag` is set equal to 1.
 - If the value of `frame_packing_arrangement_type` of the frame packing arrangement SEI message is equal to 3, `TopBottomFlag` is set equal to 0, `SideBySideFlag` is set equal to 1, `HorDiv1` is set equal to 2 and `VerDiv1` is set equal to 1.

- Otherwise, if the value of `frame_packing_arrangement_type` of the frame packing arrangement SEI message is equal to 4, `TopBottomFlag` is set equal to 1, `SideBySideFlag` is set equal to 0, `HorDiv1` is set equal to 1, and `VerDiv1` is set equal to 2.
- Otherwise, `TopBottomFlag` is set equal to 0, `SideBySideFlag` is set equal to 0, `HorDiv1` is set equal to 1, and `VerDiv1` is set equal to 1.
- If a region-wise packing SEI message with `rwpcancel_flag` equal to 0 that applies to the picture is not present, `RegionWisePackingFlag` is set equal to 0, and `ConstituentPicWidth` and `ConstituentPicHeight` are set to be equal to `cropPicWidth / HorDiv1` and `cropPicHeight / VerDiv1`, respectively, where `cropPicWidth` and `cropPicHeight` are the width and height, respectively, of the cropped decoded picture.
- Otherwise, `RegionWisePackingFlag` is set equal to 1, and `ConstituentPicWidth` and `ConstituentPicHeight` are set equal to `proj_picture_width / HorDiv1` and `proj_picture_height / VerDiv1`, respectively.

To remap colour sample locations of a fisheye video picture to a unit sphere, the sample locations in each of the active regions is converted to locations on the unit sphere as specified in clause D.3.41.7.7.

D.3.41.7.2 Projection for one sample location

Inputs to this process are:

- `pictureWidth` and `pictureHeight`, which are the width and height, respectively, of a monoscopic projected luma picture, in relative projected picture sample units, and
- the centre point of a sample location (`hPos`, `vPos`) along the horizontal and vertical axes, respectively, in relative projected picture sample units, where `hPos` and `vPos` may have non-integer real values.

Outputs of this process are:

- sphere coordinates (ϕ , θ) for the sample location in degrees relative to the local coordinate axes.

The projection for a sample location is derived as follows:

- If `ErpFlag` is equal to 1, the following applies:

- If `RegionWisePackingFlag` is equal to 0 and `erp_guard_band_flag` is equal to 1, the following applies:

$$\begin{aligned} hPos' &= hPos - \text{erp_left_guard_band_width} \\ \text{pictureWidth} &= \text{pictureWidth} - \text{erp_left_guard_band_width} - \text{erp_right_guard_band_width} \end{aligned} \quad (\text{D-54})$$

- Otherwise, the following applies:

$$hPos' = hPos \quad (\text{D-55})$$

- The following applies:

$$\begin{aligned} \phi &= 180 - hPos' * (360 \div \text{pictureWidth}) \\ \theta &= 90 - vPos * (180 \div \text{pictureHeight}) \end{aligned} \quad (\text{D-56})$$

- Otherwise (`CmpFlag` is equal to 1), it is a requirement of bitstream conformance that `pictureWidth` shall be a multiple of 3 and `pictureHeight` shall be a multiple of 2, and that `pictureWidth / 3` shall be equal to `pictureHeight / 2`, and the following applies:

```
lw = pictureWidth / 3
lh = pictureHeight / 2
w = Floor( hPos ÷ lw )
h = Floor( vPos ÷ lh )
tmpHorVal = hPos - w * lw
tmpVerVal = vPos - h * lh
hPos' = -( 2 * tmpHorVal ÷ lw ) + 1
vPos' = -( 2 * tmpVerVal ÷ lh ) + 1
if( w == 1 && h == 0 ) { /* positive x front face */
    x = 1.0
    y = hPos'
    z = vPos'
} else if( w == 1 && h == 1 ) { /* negative x back face */
    x = -1.0
```

$$\begin{aligned}
& y = -vPos' \\
& z = -hPos' \\
& \} \text{ else if(} w == 2 \ \&\& \ h == 1) \{ \text{ /* positive z top face */} \\
& \quad x = -hPos' \\
& \quad y = -vPos' \\
& \quad z = 1.0 \\
& \} \text{ else if(} w == 0 \ \&\& \ h == 1) \{ \text{ /* negative z bottom face */} \\
& \quad x = hPos' \\
& \quad y = -vPos' \\
& \quad z = -1.0 \\
& \} \text{ else if(} w == 0 \ \&\& \ h == 0) \{ \text{ /* positive y left face */} \\
& \quad x = -hPos' \\
& \quad y = 1.0 \\
& \quad z = vPos' \\
& \} \text{ else } \{ \text{ /* (} w == 2 \ \&\& \ h == 0), \text{ /* negative y right face */} \\
& \quad x = hPos' \\
& \quad y = -1.0 \\
& \quad z = vPos' \\
& \} \\
& \phi = \text{Atan2}(y, x) * 180 \div \pi \\
& \theta = \text{Asin}(z \div \text{Sqrt}(x^2 + y^2 + z^2)) * 180 \div \pi
\end{aligned} \tag{D-57}$$

D.3.41.7.3 Conversion from the local coordinate axes to the global coordinate axes

Inputs to this process are:

- rotation_yaw (α_d), rotation_pitch (β_d), rotation_roll (γ_d), all in units of degrees, and
- sphere coordinates (ϕ_d, θ_d) relative to the local coordinate axes.

Outputs of this process are:

- sphere coordinates (ϕ', θ') relative to the global coordinate axes.

The outputs are derived as follows:

$$\begin{aligned}
\phi &= \phi_d * \pi \div 180 \\
\theta &= \theta_d * \pi \div 180 \\
\alpha &= \alpha_d * \pi \div 180 \\
\beta &= \beta_d * \pi \div 180 \\
\gamma &= \gamma_d * \pi \div 180 \\
x_1 &= \text{Cos}(\phi) * \text{Cos}(\theta) \\
y_1 &= \text{Sin}(\phi) * \text{Cos}(\theta) \\
z_1 &= \text{Sin}(\theta) \\
x_2 &= \text{Cos}(\beta) * \text{Cos}(\alpha) * x_1 - \text{Cos}(\beta) * \text{Sin}(\alpha) * y_1 + \text{Sin}(\beta) * z_1 \\
y_2 &= (\text{Cos}(\gamma) * \text{Sin}(\alpha) + \text{Sin}(\gamma) * \text{Sin}(\beta) * \text{Cos}(\alpha)) * x_1 + \\
&\quad (\text{Cos}(\gamma) * \text{Cos}(\alpha) - \text{Sin}(\gamma) * \text{Sin}(\beta) * \text{Sin}(\alpha)) * y_1 - \\
&\quad \text{Sin}(\gamma) * \text{Cos}(\beta) * z_1 \\
z_2 &= (\text{Sin}(\gamma) * \text{Sin}(\alpha) - \text{Cos}(\gamma) * \text{Sin}(\beta) * \text{Cos}(\alpha)) * x_1 + \\
&\quad (\text{Sin}(\gamma) * \text{Cos}(\alpha) + \text{Cos}(\gamma) * \text{Sin}(\beta) * \text{Sin}(\alpha)) * y_1 + \\
&\quad \text{Cos}(\gamma) * \text{Cos}(\beta) * z_1 \\
\phi' &= \text{Atan2}(y_2, x_2) * 180 \div \pi \\
\theta' &= \text{Asin}(z_2) * 180 \div \pi
\end{aligned} \tag{D-58}$$

D.3.41.7.4 Conversion of sample locations for rectangular region-wise packing

Inputs to this process are:

- sample location (x, y) within the packed region, where x and y are in relative packed picture sample units, while the sample location is at an integer sample location within the packed picture,
- the width and the height (projRegWidth, projRegHeight) of the projected region, in relative projected picture sample units,

- the width and the height (packedRegWidth, packedRegHeight) of the packed region, in relative packed picture sample units,
- transform type (transformType), and
- offset values for the sampling position (offsetX, offsetY) in the range of 0, inclusive, to 1, exclusive, in horizontal and vertical relative packed picture sample units, respectively.

NOTE – offsetX and offsetY both equal to 0.5 indicate a sampling position that is in the centre point of a sample in packed picture sample units.

Outputs of this process are:

- the centre point of the sample location (hPos, vPos) within the projected region in relative projected picture sample units, where hPos and vPos may have non-integer real values.

The outputs are derived as follows:

```

if( transformType == 0 || transformType == 1 || transformType == 2 || transformType ==
  3 ) {
    horRatio = projRegWidth ÷ packedRegWidth
    verRatio = projRegHeight ÷ packedRegHeight
} else if( transformType == 4 || transformType == 5 || transformType == 6 ||
  transformType == 7 ) {
    horRatio = projRegWidth ÷ packedRegHeight
    verRatio = projRegHeight ÷ packedRegWidth
}
if( transformType == 0 ) {
    hPos = horRatio * ( x + offsetX )
    vPos = verRatio * ( y + offsetY )
} else if( transformType == 1 ) {
    hPos = horRatio * ( packedRegWidth - x - offsetX )
    vPos = verRatio * ( y + offsetY )
} else if( transformType == 2 ) {
    hPos = horRatio * ( packedRegWidth - x - offsetX )
    vPos = verRatio * ( packedRegHeight - y - offsetY )
} else if( transformType == 3 ) {
    hPos = horRatio * ( x + offsetX )
    vPos = verRatio * ( packedRegHeight - y - offsetY )
} else if( transformType == 4 ) {
    hPos = horRatio * ( y + offsetY )
    vPos = verRatio * ( x + offsetX )
} else if( transformType == 5 ) {
    hPos = horRatio * ( y + offsetY )
    vPos = verRatio * ( packedRegWidth - x - offsetX )
} else if( transformType == 6 ) {
    hPos = horRatio * ( packedRegHeight - y - offsetY )
    vPos = verRatio * ( packedRegWidth - x - offsetX )
} else if( transformType == 7 ) {
    hPos = horRatio * ( packedRegHeight - y - offsetY )
    vPos = verRatio * ( x + offsetX )
}

```

(D-59)

D.3.41.7.5 Mapping of luma sample locations within a cropped decoded picture to sphere coordinates relative to the global coordinate axes

This clause specifies the mapping of luma sample locations within a cropped decoded picture to sphere coordinates relative to the global coordinate axes.

offsetX is set equal to 0.5 and offsetY is set equal to 0.5.

If RegionWisePackingFlag is equal to 1, the following applies for each packed region n in the range of 0 to NumPackedRegions – 1, inclusive:

- For each sample location (xPackedPicture, yPackedPicture) belonging to the n-th packed region, the following applies:
 - The corresponding sample location (xProjPicture, yProjPicture) of the projected picture is derived as follows:
 - x is set equal to $x_{\text{PackedPicture}} - \text{PackedRegionLeft}[n]$.
 - y is set equal to $y_{\text{PackedPicture}} - \text{PackedRegionTop}[n]$.
 - Clause D.3.41.7.4 is invoked with x, y, PackedRegionWidth[n], PackedRegionHeight[n], ProjRegionWidth[n], ProjRegionHeight[n], TransformType[n], offsetX and offsetY as inputs, and the output is assigned to sample location (hPos, vPos).
 - xProjPicture is set equal to $\text{ProjRegionLeft}[n] + h\text{Pos}$.
 - When StereoFlag is equal to 0 or TopBottomFlag is equal to 1, and when xProjPicture is greater than or equal to proj_picture_width, xProjPicture is set equal to $x_{\text{ProjPicture}} - \text{proj_picture_width}$.
 - When SideBySideFlag is equal to 1, the following applies:
 - When $\text{ProjRegionLeft}[n]$ is less than $\text{proj_picture_width} / 2$ and xProjPicture is greater than or equal to $\text{proj_picture_width} / 2$, xProjPicture is set equal to $x_{\text{ProjPicture}} - \text{proj_picture_width} / 2$.
 - When $\text{ProjRegionLeft}[n]$ is greater than or equal to $\text{proj_picture_width} / 2$ and xProjPicture is greater than or equal to $\text{proj_picture_width}$, xProjPicture is set equal to $x_{\text{ProjPicture}} - \text{proj_picture_width} / 2$.
 - yProjPicture is set equal to $\text{ProjRegionTop}[n] + v\text{Pos}$.
 - Clause D.3.41.7.6 is invoked with xProjPicture, yProjPicture, ConstituentPicWidth, and ConstituentPicHeight as inputs, and the outputs indicating the sphere coordinates and the constituent picture index (for frame-packed stereoscopic video) for the luma sample location (xPackedPicture, yPackedPicture) belonging to the n-th packed region in the decoded picture.

Otherwise (RegionWisePackingFlag is equal 0), the following applies for each sample location (x, y) that is not an equirectangular projection guard band sample within the cropped decoded picture, where a sample location (x, y) is an equirectangular projection guard band sample when and only when ErpFlag is equal to 1, x is in the range of 0 to $\text{erp_left_guard_band_width} - 1$, inclusive, or $\text{ConstituentPicWidth} - \text{erp_right_guard_band_width}$ to $\text{ConstituentPicWidth} - 1$, inclusive, and y is in the range of 0 to $\text{ConstituentPicHeight} - 1$, inclusive:

- xProjPicture is set equal to $x + \text{offsetX}$.
- yProjPicture is set equal to $y + \text{offsetY}$.
- If ErpFlag is equal to 0, projPicWidth is set equal to ConstituentPicWidth. Otherwise (ErpFlag is equal to 1), projPicWidth is set equal to $\text{ConstituentPicWidth} - (\text{erp_left_guard_band_width} + \text{erp_right_guard_band_width})$.
- Clause D.3.41.7.6 is invoked with xProjPicture, yProjPicture, projPicWidth, and ConstituentPicHeight as inputs, and the outputs indicating the sphere coordinates and the constituent picture index (for frame-packed stereoscopic video) for the sample location (x, y) within the region-wise packed picture.

D.3.41.7.6 Conversion from a sample location in a projected picture to sphere coordinates relative to the global coordinate axes

Inputs to this process are:

- the centre point of a sample location (xProjPicture, yProjPicture) within a projected picture, where xProjPicture and yProjPicture are in relative projected picture sample units and may have non-integer real values, and
- pictureWidth and pictureHeight, which are the width and height, respectively, of a monoscopic projected luma picture, in relative projected picture sample units.

Outputs of this process are:

- sphere coordinates (azimuthGlobal, elevationGlobal), in units of degrees relative to the global coordinate axes, and
- when StereoFlag is equal to 1, the index of the constituent picture (constituentPicture) equal to 0 or 1.

The outputs are derived with the following ordered steps:

1. constituentPicture, xProjPicture, and yProjPicture are conditionally set as follows:
 - If xProjPicture is greater than or equal to pictureWidth or yProjPicture is greater than or equal to pictureHeight, the following applies:

- constituentPicture is set equal to 1.
 - When xProjPicture is greater than or equal to pictureWidth, xProjPicture is set to xProjPicture – pictureWidth.
 - When yProjPicture is greater than or equal to pictureHeight, yProjPicture is set to yProjPicture – pictureHeight.
 - Otherwise, constituentPicture is set equal to 0.
2. Clause D.3.41.7.2 is invoked with pictureWidth, pictureHeight, xProjPicture, and yProjPicture as inputs, and the output is assigned to azimuthLocal, elevationLocal.
3. azimuthGlobal and elevationGlobal are set as follows:
- If RotationFlag is equal to 1, clause D.3.41.7.3 is invoked with azimuthLocal, elevationLocal, RotationYaw, RotationPitch, and RotationRoll as inputs, and the output is assigned to azimuthGlobal and elevationGlobal.
 - Otherwise, azimuthGlobal is set equal to azimuthLocal and elevationGlobal is set equal to elevationLocal.

D.3.41.7.7 Conversion from a sample location of an active area to sphere coordinates relative to the global coordinate axes

Inputs to this process are:

- the sample location (x, y) in units of luma samples,
- the centre location (x_c, y_c) and the radius (r_c) of the circular region that contains the i-th active area, given by fisheye_circular_region_centre_x[i], fisheye_circular_region_centre_y[i], and fisheye_circular_region_radius[i], respectively, all in units of 2⁻¹⁶ luma samples,
- the field of view (θ_v) of the lens corresponding to the i-th active area, given by fisheye_field_of_view[i], in units of 2⁻¹⁶ degrees,
- the rotation parameters (α_c, β_c, γ_c), given by fisheye_camera_centre_azimuth[i], fisheye_camera_centre_elevation[i], and fisheye_camera_centre_tilt[i], respectively, all in units of 2⁻¹⁶ degrees, and
- the number of polynomial coefficients numCoeffs and the polynomial coefficients coeffVal[j] (for j ranging from 0 to numCoeffs – 1, inclusive) of the i-th active area, given by fisheye_num_polynomial_coeffs[i] and fisheye_polynomial_coeff[i][j] (for j ranging from 0 to fisheye_num_polynomial_coeffs[i] – 1, inclusive), respectively.

Outputs of this process are:

- sphere coordinates (φ, θ) relative to the global coordinate axes.

The method of converting a sample location of an active area to sphere coordinates is determined as follows:

- If numCoeffs is equal to 0, there is only one method of converting a sample location of an active area to sphere coordinates that is specified, which is to not use polynomial coefficients.
- Otherwise (numCoeffs is not equal to 0), there are two methods of converting a sample location of an active area to sphere coordinates that are specified, which are to not use polynomial coefficients or to use polynomial coefficients. The method using polynomial coefficients is preferred, as this method is intended to provide a more precise model of the fisheye characteristics. However, the other method may also be appropriate for some uses, as it provides a single conversion process that can be used regardless of whether numCoeffs is equal to 0 or not. This Specification does not prescribe which of the two methods is to be used in this case.

The outputs are derived as follows:

- If polynomial coefficients are not used, the angle φ' is derived by

$$\phi' = (\text{Sqrt}((x - x_c \div 2^{16})^2 + (y - y_c \div 2^{16})^2) \div (r_c \div 2^{16})) * (\theta_v \div 2^{16} * \pi \div 180) \div 2 \quad (\text{D-60})$$

- Otherwise (polynomial coefficients are used), the angle φ' is derived by

$$\phi' = \sum_{j=0}^{\text{numCoeffs}-1} ((\text{coeffVal}[j] * 2^{-24}) * (\text{Sqrt}((x - x_c * 2^{-16})^2 + (y - y_c * 2^{-16})^2) \div (r_c * 2^{-16}))^j) \quad (\text{D-61})$$

The outputs are then derived as follows:

$$\begin{aligned}
\theta' &= \text{Atan2}(y - y_c \div 2^{16}, x - x_c \div 2^{16}) \\
x_1 &= \text{Cos}(\phi') \\
y_1 &= \text{Sin}(\phi') * \text{Cos}(\theta') \\
z_1 &= \text{Sin}(\phi') * \text{Sin}(\theta') \\
\alpha &= (\alpha_c \div 2^{16}) * \pi \div 180 \\
\beta &= (\beta_c \div 2^{16}) * \pi \div 180 \\
\gamma &= (\gamma_c \div 2^{16}) * \pi \div 180 \\
x_2 &= \text{Cos}(\beta) * \text{Cos}(\gamma) * x_1 - \text{Cos}(\beta) * \text{Sin}(\gamma) * y_1 + \text{Sin}(\beta) * z_1 \\
y_2 &= (\text{Cos}(\alpha) * \text{Sin}(\gamma) + \text{Sin}(\alpha) * \text{Sin}(\beta) * \text{Cos}(\gamma)) * x_1 + \\
&\quad (\text{Cos}(\alpha) * \text{Cos}(\gamma) - \text{Sin}(\alpha) * \text{Sin}(\beta) * \text{Sin}(\gamma)) * y_1 - \\
&\quad \text{Sin}(\alpha) * \text{Cos}(\beta) * z_1 \\
z_2 &= (\text{Sin}(\alpha) * \text{Sin}(\gamma) - \text{Cos}(\alpha) * \text{Sin}(\beta) * \text{Cos}(\gamma)) * x_1 + \\
&\quad (\text{Sin}(\alpha) * \text{Cos}(\gamma) + \text{Cos}(\alpha) * \text{Sin}(\beta) * \text{Sin}(\gamma)) * y_1 + \\
&\quad \text{Cos}(\alpha) * \text{Cos}(\beta) * z_1 \\
\phi &= \text{Atan2}(y_2, x_2) * 180 \div \pi \\
\theta &= \text{Asin}(z_2) * 180 \div \pi
\end{aligned} \tag{D-62}$$

D.3.42 Regional nesting SEI message semantics

The regional nesting SEI message provides a mechanism to associate SEI messages with regions of the picture. The associated SEI messages are conveyed within the regional nesting SEI message.

A regional nesting SEI message contains one or more SEI messages. When an SEI message is contained in a regional nesting SEI message, the contained SEI message is referred to as a region-nested SEI message. When an SEI message is not contained in a regional nesting SEI message, the SEI message is referred to as a non-region-nested SEI message.

For each region-nested SEI message in a regional nesting SEI message, one or more regions are specified in the regional nesting SEI message, and the semantics of the region-nested SEI message are to be interpreted as applying to each of these regions.

The list `listOfRegionNestableMessageTypes` includes the following types of SEI messages:

- Film grain characteristics SEI message,
- Post filter hint SEI message,
- Tone mapping information SEI message identified with a particular value of `tone_map_id`,
- Chroma resampling filter hint SEI message,
- Knee function information SEI message identified with a particular value of `knee_function_id`,
- Colour remapping information SEI message identified with a particular value of `colour_remap_id`,
- Content colour volume SEI message.

NOTE 1 – SEI messages of each of the following are considered different types of SEI messages: 1) tone mapping information SEI messages with different values of `tone_map_id`, 2) knee function information SEI messages with different values of `knee_function_id`, and 3) colour remapping information SEI messages with different values of `colour_remap_id`.

When an SEI message of a particular type in `listOfRegionNestableMessageTypes` has `film_grain_characteristics_cancel_flag`, `tone_map_cancel_flag`, `knee_function_cancel_flag`, or `colour_remap_cancel_flag` equal to 1, regardless of whether it is region-nested or non-region-nested, it cancels the persistence of all the region-nested SEI messages of that type, regardless of their associated regions. When an SEI message of a particular type having `film_grain_characteristics_persistence_flag`, `tone_map_persistence_flag`, `knee_function_persistence_flag`, or `colour_remap_persistence_flag` equal to 1 is region-nested, the persistence of the SEI message is determined by the semantics of the SEI message, irrespective of which region it applies to.

NOTE 2 – A region-nested SEI message has the same persistence scope as if the SEI message was non-region-nested.

NOTE 3 – A region-nested SEI message does not cancel the persistence of a non-region-nested SEI message of the same type.

The list `listOfAllowedRegionalNestableMessageTypes` includes all the entries in the list `listOfRegionNestableMessageTypes` and also the following additional types of SEI messages:

- User data registered by Rec. ITU-T T.35 SEI message,

- User data unregistered SEI message.

In bitstreams conforming to this version of this Specification, the regional nesting SEI message shall not contain any SEI message that is not in `listOfAllowedRegionNestableMessageTypes`. Decoders encountering a region-nested SEI message that does not belong to `listOfAllowedRegionNestableMessageTypes` shall ignore the region-nested SEI message.

When an access unit contains both region-nested SEI messages of a particular type in `listOfRegionNestableMessageTypes` and non-region-nested SEI messages of the same type, decoders shall ignore either all the region-nested SEI message of that type or all the non-region-nested SEI messages of that type. Unless indicated otherwise by some means not specified in this Specification, when an access unit contains both region-nested SEI messages of a particular type in `listOfRegionNestableMessageTypes` and non-region-nested SEI messages of the same type, the region-nested SEI messages should be preferred to be considered as applicable to the access unit.

A region-nested SEI messages should not be extracted and sent as a non-region-nested SEI message, as the values signalled in the region-nested SEI message may not be applicable outside the indicated regions.

regional_nesting_id contains an identifying number that may be used to identify the purpose of the one or more SEI messages that are region-nested in the regional nesting SEI message. The value of `regional_nesting_id` shall be in the range of 0 to $2^{16} - 1$, inclusive.

Values of `regional_nesting_id` from 0 to 255, inclusive, and from 512 to $2^{15} - 1$, inclusive, may be used as determined by the application. Values of `regional_nesting_id` from 256 to 511, inclusive, and from 2^{15} to $2^{16} - 1$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of `regional_nesting_id` in the range of 256 to 511, inclusive, or in the range of 2^{15} to $2^{16} - 1$, inclusive, shall ignore it.

regional_nesting_num_rect_regions specifies the number of rectangular regions specified in the regional nesting SEI message. The value of `regional_nesting_num_rect_regions` shall be in the range of 1 to 255, inclusive. The value of `regional_nesting_num_rect_regions` equal to 0 is reserved for future use by ITU-T | ISO/IEC and shall not be used in bitstreams conforming to this version of this Specification. Decoders shall ignore regional nesting SEI messages with `regional_nesting_num_rect_regions` equal to 0.

regional_nesting_rect_region_id[i] specifies the identifier for the i-th rectangular region specified in the regional nesting SEI message.

Unless indicated otherwise by some means not specified in this Specification, when a sample belongs to more than one region indicated as applying to more than one region-nested SEI message of a particular type in `listOfRegionNestableMessageTypes`, among these region-nested SEI messages, only those that are associated with the region that has the greatest value of `regional_nesting_rect_region_id[i]` are considered as applying to the sample, while the rest of these region-nested SEI messages are considered as not applying to the sample.

NOTE 4 – When there are more than one of these region-nested SEI messages associated with the region that has the greatest value of `regional_nesting_rect_region_id[i]`, they are identical per other expressed constraints.

It is a requirement of bitstream conformance that the value of `regional_nesting_rect_region_id[i]` shall not be the same for any two different values of i in the range of 0 to `regional_nesting_num_rect_regions` – 1, inclusive, in the regional nesting SEI message.

When a region-nested SEI message of a particular type in `listOfRegionNestableMessageTypes` is indicated as applying to a list of regions `listA` in the current picture and another region-nested SEI message of the same type is indicated as applying to another list of regions `listB` in the current picture, it is a requirement of bitstream conformance that, for any pair of regions formed by choosing one from `listA` and the other from `listB`, the value of `regional_nesting_rect_region_id[i]` of the two regions shall not be the same unless the two regions are identical (i.e., both position and size are the same) and the two region-nested SEI messages are identical.

regional_nesting_rect_left_offset[i], **regional_nesting_rect_right_offset[i]**, **regional_nesting_rect_top_offset[i]**, and **regional_nesting_rect_bottom_offset[i]** specify the coordinates of the i-th rectangular region specified in the SEI message. The offsets for the rectangular region are specified in units of luma samples. The i-th rectangular region contains the luma samples with horizontal picture coordinates from `SubWidthC * regional_nesting_rect_left_offset[i]` to `pic_width_in_luma_samples – (SubWidthC * regional_nesting_rect_right_offset[i] + 1)`, inclusive, and vertical picture coordinates from `SubHeightC * regional_nesting_rect_top_offset[i]` to `pic_height_in_luma_samples – (SubHeightC * regional_nesting_rect_bottom_offset[i] + 1)`, inclusive.

The value of `SubWidthC * (regional_nesting_rect_left_offset[i] + regional_nesting_rect_right_offset[i])` shall be less than `pic_width_in_luma_samples` and the value of `SubHeightC * (regional_nesting_rect_top_offset[i] + regional_nesting_rect_bottom_offset[i])` shall be less than `pic_height_in_luma_samples`.

num_sei_messages_in_regional_nesting_minus1 plus 1 specifies the number of region-nested SEI messages in the regional nesting SEI message. The value of `num_sei_messages_in_regional_nesting_minus1` shall be in the range of 0 to 255, inclusive.

num_regions_for_sei_message[*i*] specifies the number of regions to which the *i*-th region-nested SEI message is associated. When **regional_nesting_num_rect_regions** is greater than 0, the value of **num_regions_for_sei_message**[*i*] shall be in the range of 0 to **regional_nesting_num_rect_regions**, inclusive. When **num_regions_for_sei_message**[*i*] is equal to 0, the *i*-th region-nested SEI message applies to all the regions specified in the regional nesting SEI message.

regional_nesting_sei_region_idx[*i*][*j*] specifies the index, into the list of regions specified in the regional nesting SEI message, of the *j*-th region to which the *i*-th region-nested SEI message in the regional nesting SEI message is associated. The value of **regional_nesting_sei_region_idx**[*i*][*j*] shall be in the range of 0 to **regional_nesting_num_rect_region** – 1, inclusive.

D.3.43 Motion-constrained tile sets extraction information sets SEI message semantics

The motion-constrained tile sets extraction information sets SEI message provides supplemental information that can be used in the motion-constrained tile set (MCTS) sub-bitstream extraction as specified below to generate a conforming bitstream for an MCTS set. The information consists of a number of extraction information sets, each defining a number of MCTS sets and containing RBSP bytes of the replacement VPSs, SPSs, and PPSs to be used during the MCTS sub-bitstream extraction process. Each extraction information set can be shared by multiple MCTS sets, i.e., is used for extraction of any of these MCTS sets.

An MCTS extraction information sets SEI message shall not be present in an access unit unless there is a temporal MCTS SEI message present in the access unit. A temporal MCTS SEI message present in the same access unit as an MCTS extraction information sets SEI message is referred to as the associated temporal MCTS SEI message of the MCTS extraction information sets SEI message.

An MCTS extraction information sets SEI message applies to the same set of pictures as the associated temporal MCTS SEI message, i.e., the **associatedPicSet** of the MCTS extraction information sets SEI message is the same as the **associatedPicSet** of the associated temporal MCTS SEI message.

When more than one MCTS extraction information sets SEI message is present for the pictures of **associatedPicSet**, these MCTS extraction information sets SEI messages shall contain identical content.

NAL units that contain tiles belonging to any particular MCTS **mctsA** shall not contain tiles that do not belong to **mctsA**.

num_info_sets_minus1 plus 1 specifies the number of extraction information sets contained in the MCTS extraction information sets SEI message. The value of **num_info_sets_minus1** shall be in the range of 0 to 2047, inclusive.

The *i*-th extraction information set is assigned an MCTS extraction information set identifier value equal to *i*.

num_mcts_sets_minus1[*i*] plus 1 specifies the number of MCTS sets that share the *i*-th extraction information set. The value of **num_mcts_sets_minus1**[*i*] shall be in the range of 0 to 2047, inclusive.

num_mcts_in_set_minus1[*i*][*j*] plus 1 specifies the number of MCTSs in the *j*-th MCTS set that is associated with the *i*-th extraction information set. The value of **num_mcts_in_set_minus1**[*i*][*j*] shall be in the range of 0 to 511, inclusive.

idx_of_mcts_in_set[*i*][*j*][*k*] specifies the MCTS index of the *k*-th MCTS in the *j*-th MCTS set that is associated with the *i*-th extraction information set. The value of **idx_of_mcts_in_set**[*i*][*j*][*k*] shall be in the range of 0 to 511, inclusive.

slice_reordering_enabled_flag[*i*] equal to 1 specifies that the MCTS sub-bitstream extraction using the *i*-th extraction information set includes reordering of extracted slice segments and that the **slice_segment_address** of the *j*-th slice segment in bitstream order associated with any of the extracted MCTS sets with *j* in the range of 0 to **num_slice_segments_minus1**[*i*] is set to **output_slice_segment_address**[*i*][*j*]. **slice_reordering_enabled_flag**[*i*] equal to 0 indicates that the MCTS sub-bitstream extraction using the *i*-th extraction information set does not include a reordering of extracted slice segments and that the **slice_segment_address** of all extracted slice segments is calculated during extraction.

num_slice_segments_minus1[*i*] plus 1 specifies the number of slice segments associated with any MCTS set of the *i*-th extraction information set when **slice_reordering_enabled_flag**[*i*] is equal to 1. The value of **num_slice_segments_minus1**[*i*] shall be in the range of 0 to 1 024, inclusive.

output_slice_segment_address[*i*][*j*] specifies the slice segment address of the *j*-th slice segment in bitstream order associated with any of the MCTS sets of the *i*-th extraction information set when **slice_reordering_enabled_flag**[*i*] is equal to 1. The length of the **output_slice_segment_address**[*i*][*j*] syntax element is **Ceil(Log2(PicSizeInCtbsY))** bits. The value of **output_slice_segment_address** shall be in the range of 0 to **PicSizeInCtbsY** – 1, inclusive and no value of **output_slice_segment_address**[*i*][*j*] shall be equal to some **output_slice_segment_address**[*i*][*k*] for *j* not equal to *k* in the extraction information set SEI message.

num_vps_in_info_set_minus1[*i*] plus 1 specifies the number of replacement VPSs in the *i*-th extraction information set. The value of **num_vps_in_info_set_minus1**[*i*] shall be in the range of 0 to 15, inclusive.

vps_rbsp_data_length[i][j] specifies the number of RBSP data bytes of the j-th replacement VPS in the i-th extraction information set.

num_sps_in_info_set_minus1[i] plus 1 specifies the number of replacement SPSs in the i-th extraction information set. The value of **num_sps_in_info_set_minus1**[i] shall be in the range of 0 to 15, inclusive.

sps_rbsp_data_length[i][j] specifies the number of RBSP data bytes of the j-th replacement SPS in the i-th extraction information set.

num_pps_in_info_set_minus1[i] plus 1 specifies the number of replacement PPSs in the i-th extraction information set. The value of **num_pps_in_info_set_minus1**[i] shall be in the range of 0 to 63, inclusive.

pps_nuh_temporal_id_plus1[i][j] minus 1 specifies the temporal identifier of the j-th replacement PPS in the i-th extraction information set.

pps_rbsp_data_length[i][j] specifies the number of RBSP data bytes of the j-th replacement PPS in the i-th extraction information set.

mcts_alignment_bit_equal_to_zero shall be equal to 0.

vps_rbsp_data_byte[i][j][k] contains the k-th byte of the RBSP data of the j-th replacement VPS in the i-th extraction information set.

sps_rbsp_data_byte[i][j][k] contains the k-th byte of the RBSP data of the j-th replacement SPS in the i-th extraction information set.

pps_rbsp_data_byte[i][j][k] contains the k-th byte of the RBSP data of the j-th replacement PPS in the i-th extraction information set.

The MCTS sub-bitstream extraction process is specified as follows:

- Let a bitstream **inBitstream**, a target MCTS set index **mctsSetIdxTarget**, a target MCTS extraction information set identifier **mctsEisIdTarget**, and a target highest TemporalId value **mctsTidTarget** be the inputs to the MCTS sub-bitstream extraction process.
- The output of the MCTS sub-bitstream extraction process is a sub-bitstream **outBitstream** derived as follows:

- The bitstream **outBitstream** is set to be identical to the bitstream **inBitstream**.
- The lists **ausWithVps**, **ausWithSps**, and **ausWithPps** are set to consist of all access units within **outBitstream** containing non-VCL NAL units with **nal_unit_type** equal to **VPS_NUT**, **SPS_NUT**, or **PPS_NUT**.
- Remove all SEI NAL units that contain non-MCTS-nested SEI messages.

NOTE 1 – A "smart" bitstream extractor might include appropriate non-MCTS-nested SEI messages in the extracted MCTS sub-bitstream, provided that the SEI messages applicable to the MCTS sub-bitstream were present as MCTS-nested SEI messages in the original bitstream.

- Remove from **outBitstream** all of the following NAL units:
 - VCL NAL units that contain tiles not belonging to any of the MCTSs with MCTS index equal to **idx_of_mcts_in_set**[**mctsEisIdTarget**][**mctsSetIdxTarget**][k] for each value of k in the range of 0 to **num_mcts_in_set_minus1**[**mctsEisIdTarget**][**mctsSetIdxTarget**], inclusive.
 - Non-VCL NAL units with **nal_unit_type** equal to **VPS_NUT**, **SPS_NUT**, or **PPS_NUT**.
- Insert into each access unit within the list **ausWithVps** in **outBitstream** **num_vps_in_info_set_minus1**[**mctsEisIdTarget**] plus 1 VPS NAL units generated from the RBSP data of the list of replacement VPSs in the **mctsEisIdTarget**-th MCTS extraction information set. For each VPS NAL unit that is generated the **nuh_layer_id** is set equal to 0 and **nuh_temporal_id_plus1** is set equal to 1.
- Insert into each access unit within the list **ausWithSps** in **outBitstream** **num_sps_in_info_set_minus1**[**mctsEisIdTarget**] plus 1 SPS NAL units generated from the RBSP data of the list of replacement SPSs in the **mctsEisIdTarget**-th MCTS extraction information set. For each SPS NAL unit that is generated the **nuh_layer_id** is set equal to 0 and **nuh_temporal_id_plus1** is set equal to 1.
- Insert into each access unit within the list **ausWithPps** in **outBitstream** PPS NAL units generated from the RBSP data of the replacement PPSs, in the **mctsEisIdTarget**-th MCTS extraction information set, for which **pps_nuh_temporal_id_plus1**[**mctsEisIdTarget**][j] is less than or equal to **mctsTidTarget**. For each PPS NAL unit that is generated the **nuh_layer_id** is set equal to 0, and for the PPS NAL unit that is generated from the RBSP data of the j-th replacement PPS in the **mctsEisIdTarget**-th MCTS extraction information set, **nuh_temporal_id_plus1** is set equal to **pps_nuh_temporal_id_plus1**[**mctsEisIdTarget**][j].

NOTE 2 – The values of `pps_pic_parameter_set_id` of the replacement PPSs should be identical to the values of `pps_pic_parameter_set_id` of the removed PPSs to retain the value of `slice_pic_parameter_set_id` in slice headers of the original bitstream.

- Remove from `outBitstream` all NAL units with `TemporalId` greater than `mctsTidTarget`.
- If `slice_reordering_enabled_flag[mctsEISIdTarget]` is equal to 0, the CTB raster and tile scanning conversion process as specified in clause 6.5.1 is invoked with the syntax element values of the replacement SPS and PPS as inputs. The output `CtbAddrRsToTs[ctbAddrRs]` is assigned to `extCtbAddrRsToTs[ctbAddrRs]` and `CtbAddrTsToRs[ctbAddrTs]` is assigned to `extCtbAddrTsToRs[ctbAddrTs]`. For each remaining VCL NAL units in `outBitstream`, adjust the slice segment header as follows:
 - For the first VCL NAL unit within each access unit, set the value of `first_slice_segment_in_pic_flag` equal to 1, and set the value of `slice_segment_address` to be equal to 0.
 - For each remaining VCL NAL units in `outBitstream`, let `ctbAddrRs` be the value of the raster scan address of the last CTB in the previous VCL NAL unit in bitstream order within a coded picture of `outBitstream`, set the value of `first_slice_segment_in_pic_flag` equal to 0, and set the value of `slice_segment_address` equal to `extCtbAddrTsToRs[extCtbAddrRsToTs[ctbAddrRs] + 1]`.
- Otherwise (`slice_reordering_enabled_flag[mctsEISIdTarget]` is equal to 1), the following applies:
 - For the k -th VCL NAL units of each access unit in `outBitstream`, set the value of `first_slice_segment_in_pic_flag` equal to 0 and `slice_segment_address` equal to `output_slice_segement_address[mctsEISIdTarget][j]`, where j is in the range of 0 to `num_slice_segments_minus1[mctsEISIdTarget]`, inclusive.
 - Reorder the VCL NAL units within each access unit for ascending values of `slice_segment_address`.
 - For the first VCL NAL unit within each access unit, set the value of `first_slice_segment_in_pic_flag` equal to 1.

NOTE 3 – The extracted MCTS sub-bitstream might have a smaller luma picture size compared to the original bitstream which might require adjustment of the length of `slice_segment_address` and the `byte_alignment()` in slice headers.

It is a requirement of bitstream conformance for the input bitstream that any output sub-bitstream that is the output of the MCTS sub-bitstream extraction process specified in this clause shall be a conforming bitstream.

D.3.44 Motion-constrained tile sets extraction information nesting SEI message semantics

The motion-constrained tile sets extraction information nesting SEI message, also referred to as the MCTS nesting SEI message, provides a mechanism to carry and associate the SEI messages with bitstream subsets corresponding to one or more MCTSs. An SEI message contained in an MCTS nesting SEI message is referred to as MCTS-nested or an MCTS-nested SEI message, and an SEI message that is not contained in an MCTS nesting SEI message is referred to as a non-MCTS-nested SEI message.

In the MCTS sub-bitstream extraction process as specified in the semantics of the MCTS extraction information sets SEI message, the MCTS-nested SEI messages applicable to the MCTSs in an MCTS set in an access unit can be included in the corresponding access unit of the extracted sub-bitstream as non-MCTS-nested SEI messages.

An MCTS-nesting SEI message shall not be present for a picture unless the picture belongs to the `associatedPicSet` of a temporal MCTS SEI message. This temporal MCTS SEI message is referred to as the associated temporal MCTS SEI message of the MCTS-nesting SEI message.

An SEI NAL unit containing an MCTS nesting SEI message shall not contain any other SEI message that is not MCTS-nested in the MCTS nesting SEI message.

all_mcts_flag equal to 0 specifies that the list `nesting_mctsid` is set to consist of the MCTS indices indicated by all instances of `idx_of_associated_mcts[i]` for i from 0 to `num_associated_mcts_minus1`, inclusive. **all_mcts_flag** equal to 1 specifies that the list `nesting_mctsid` consists of the MCTS indices of all the MCTSs indicated by the associated temporal MCTS SEI message.

The MCTS-nested SEI messages apply to all MCTSs for which the MCTS indices are included in the list `nesting_mctsid`.

num_associated_mcts_minus1 plus 1 specifies the number of the following MCTS indices. The value of `num_associated_mcts_minus1` shall be in the range of 0 to 511, inclusive.

idx_of_associated_mcts[i] indicates the MCTS index of the i -th MCTS associated with the following MCTS-nested SEI messages. The value of `idx_of_associated_mcts[i]` shall be in the range of 0 to 511, inclusive.

num_sei_messages_in_mcts_extraction_nesting_minus1 plus 1 specifies the number of the following MCTS-nested SEI messages.

mcts_nesting_zero_bit shall be equal to 0.

D.3.45 SEI manifest SEI message semantics

The SEI manifest SEI message conveys information on SEI messages that are indicated as expected (i.e., likely) to be present or not present. Such information may include the following:

- The indication that certain types of SEI messages are expected (i.e., likely) to be present (although not guaranteed to be present) in the CVS.
- For each type of SEI message that is indicated as expected (i.e., likely) to be present in the CVS, the degree of expressed necessity of interpretation of the SEI messages of this type, as follows:
 - The degree of necessity of interpretation of an SEI message type may be indicated as "necessary", "unnecessary", or "undetermined".
 - An SEI message is indicated by the encoder (i.e., the content producer) as being "necessary" when the information conveyed by the SEI message is considered as necessary for interpretation by the decoder or receiving system in order to properly process the content and enable an adequate user experience; it does not mean that the bitstream is required to contain the SEI message in order to be a conforming bitstream. It is at the discretion of the encoder to determine which SEI messages are to be considered as necessary in a particular CVS. However, it is suggested that some SEI messages, such as the frame packing arrangement, segmented rectangular frame packing arrangement, and omnidirectional projection indication SEI messages, should typically be considered as necessary.
- The indication that certain types of SEI messages are expected (i.e., likely) not to be present (although not guaranteed not to be present) in the CVS.

NOTE – An example of such a usage of an SEI manifest SEI message is to express the expectation that there are no frame packing arrangement SEI messages, segmented rectangular frame packing arrangement SEI messages, display orientation SEI messages, or omnidirectional projection indication SEI messages in the CVS, and therefore that the rendering of the decoded video pictures for display purposes would not need any of the additional post-processing that is commonly associated with the interpretation of these SEI messages.

The content of an SEI manifest SEI message may, for example, be used by transport-layer or systems-layer processing elements to determine whether the CVS is suitable for delivery to a receiving and decoding system, based on whether the receiving system can properly process the CVS to enable an adequate user experience or whether the CVS satisfies the application needs.

When an SEI manifest SEI message is present in any access unit of a CVS, an SEI manifest SEI message shall be present in the first access unit of the CVS. The SEI manifest SEI message persists in decoding order from the current access unit until the end of the CVS. When there are multiple SEI manifest SEI messages present in a CVS, they shall have the same content.

An SEI NAL unit containing an SEI manifest SEI message shall not contain any other SEI messages other than SEI prefix indication SEI messages. When present in an SEI NAL unit, the SEI manifest SEI message shall be the first SEI message in the SEI NAL unit.

manifest_num_sei_msg_types specifies the number of types of SEI messages for which information is provided in the SEI manifest SEI message.

manifest_sei_payload_type[i] indicates the payloadType value of the i-th type of SEI message for which information is provided in the SEI manifest SEI message. The values of **manifest_sei_payload_type[m]** and **manifest_sei_payload_type[n]** shall not be identical when m is not equal to n.

manifest_sei_description[i] provides information on SEI messages with payloadType equal to **manifest_sei_payload_type[i]** as specified in Table D.23.

Table D.23 – Interpretation of manifest_sei_description[i]

Value	Description
0	Indicates that there is no SEI message with payloadType equal to manifest_sei_payload_type[i] expected to be present in the CVS.
1	Indicates that there are SEI messages with payloadType equal to manifest_sei_payload_type[i] expected to be present in the CVS, and these SEI messages are considered as necessary.
2	Indicates that there are SEI messages with payloadType equal to manifest_sei_payload_type[i] expected to be present in the CVS, and these SEI messages are considered as unnecessary.

3	Indicates that there are SEI messages with payloadType equal to manifest_sei_payload_type[i] expected to be present in the CVS, and the necessity of these SEI messages is undetermined.
4..255	Reserved

The value of manifest_sei_description[i] shall be in the range of 0 to 3, inclusive, in bitstreams conforming to this version of this Specification. Other values for manifest_sei_description[i] are reserved for future use by ITU-T | ISO/IEC. Decoders shall allow the value of manifest_sei_description[i] greater than or equal to 4 to appear in the syntax and shall ignore all information for payloadType equal to manifest_sei_payload_type[i] signalled in the SEI manifest SEI message and shall ignore all SEI prefix indication SEI messages with prefix_sei_payload_type equal to manifest_sei_payload_type[i] when manifest_sei_description[i] is greater than or equal to 4.

D.3.46 SEI prefix indication SEI message semantics

The SEI prefix indication SEI message carries one or more SEI prefix indications for SEI messages of a particular value of payloadType. Each SEI prefix indication is a bit string that follows the SEI payload syntax of that value of payloadType and contains a number of complete syntax elements starting from the first syntax element in the SEI payload.

Each SEI prefix indication for an SEI message of a particular value of payloadType indicates that one or more SEI messages of this value of payloadType are expected (i.e., likely) to be present in the CVS and to start with the provided bit string. A starting bit string would typically contain only a true subset of an SEI payload of the type of SEI message indicated by the payloadType, may contain a complete SEI payload, and shall not contain more than a complete SEI payload. It is not prohibited for SEI messages of the indicated value of payloadType to be present that do not start with any of the indicated bit strings.

These SEI prefix indications should provide sufficient information for indicating what type of processing is needed or what type of content is included. The former (type of processing) indicates decoder-side processing capability, e.g., whether some type of frame unpacking is needed. The latter (type of content) indicates, for example, whether the bitstream contains subtitle captions in a particular language.

The content of an SEI prefix indication SEI message may, for example, be used by transport-layer or systems-layer processing elements to determine whether the CVS is suitable for delivery to a receiving and decoding system, based on whether the receiving system can properly process the CVS to enable an adequate user experience or whether the CVS satisfies the application needs (as determined in some manner by external means outside the scope of this Specification).

In one example, when the payloadType indicates the frame packing arrangement SEI message, an SEI prefix indication should include up to at least the syntax element frame_packing_arrangement_type; and when the payloadType indicates the omnidirectional projection indication SEI message, an SEI prefix indication should include up to at least the syntax element projection_type.

In another example, for user data registered SEI messages that are used to carry captioning information, an SEI prefix indication should include up to at least the language code; and for user data unregistered SEI messages extended for private use, an SEI prefix indication should include up to at least the UUID.

When an SEI prefix indication SEI message is present in any access unit of a CVS, an SEI prefix indication SEI message shall be present in the first access unit of the CVS. The SEI prefix indication SEI message persists in decoding order from the current access unit until the end of the CVS. When there are multiple SEI prefix indication SEI messages present in a CVS for a particular value of payloadType, they shall have the same content.

An SEI NAL unit containing an SEI prefix indication SEI message for a particular value of payloadType shall not contain any other SEI messages other than an SEI manifest SEI message and SEI prefix indication SEI messages for other values of payloadType.

prefix_sei_payload_type indicates the payloadType value of the SEI messages for which one or more SEI prefix indications are provided in the SEI prefix indication SEI message. When an SEI manifest SEI message is also present for the CVS, the value of prefix_sei_payload_type shall be equal to one of the manifest_sei_payload_type[m] values for which manifest_sei_description[m] is equal to 1 to 3, inclusive, as indicated by an SEI manifest SEI message that applies to the CVS.

num_sei_prefix_indications_minus1 plus 1 specifies the number of SEI prefix indications.

num_bits_in_prefix_indication_minus1[i] plus 1 specifies the number of bits in the i-th SEI prefix indication.

sei_prefix_data_bit[i][j] specifies the j-th bit of the i-th SEI prefix indication.

The bits sei_prefix_data_bit[i][j] for j ranging from 0 to num_bits_in_prefix_indication_minus1[i], inclusive, follow the syntax of the SEI payload with payloadType equal to prefix_sei_payload_type, and contain a number of complete

syntax elements starting from the first syntax element in the SEI payload syntax, and may or may not contain all the syntax elements in the SEI payload syntax. The last bit of these bits (i.e., the bit `sei_prefix_data_bit[i][num_bits_in_prefix_indication_minus1[i]]`) shall be the last bit of a syntax element in the SEI payload syntax, unless it is a bit within an `itu_t_t35_payload_byte` or `user_data_payload_byte`.

NOTE – The exception for `itu_t_t35_payload_byte` and `user_data_payload_byte` is provided because these syntax elements may contain externally-specified syntax elements, and the determination of the boundaries of such externally-specified syntax elements is a matter outside the scope of this Specification.

`byte_alignment_bit_equal_to_one` shall be equal to 1.

D.3.47 Annotated regions SEI message semantics

The annotated regions SEI message carries parameters that identify annotated regions using bounding boxes representing the size and location of identified objects.

`ar_cancel_flag` equal to 1 indicates that the SEI message cancels the persistence of any previous annotated regions SEI message that is associated with one or more layers to which the annotated regions SEI message applies. **`ar_cancel_flag`** equal to 0 indicates that annotated regions information follows.

When **`ar_cancel_flag`** equal to 1 or a new CLVS of the current layer begins, the variables `LabelAssigned[i]`, `ObjectTracked[i]`, and `ObjectBoundingBoxAvail` are set equal to 0 for `i` in the range of 0 to 255, inclusive.

Let `picA` be the current picture. Each region identified in the annotated regions SEI message persists for the current layer in output order until any of the following conditions are true:

- A new CLVS of the current layer begins.
- The bitstream ends.
- A picture `picB` in the current layer in an access unit containing an annotated regions SEI message that is applicable to the current layer is output for which `PicOrderCnt(picB)` is greater than `PicOrderCnt(picA)`, where `PicOrderCnt(picB)` and `PicOrderCnt(picA)` are the `PicOrderCntVal` values of `picB` and `picA`, and the semantics of the annotated regions SEI message for `PicB` cancels the persistence of the region identified in the annotated regions SEI message for `PicA`.

`ar_not_optimized_for_viewing_flag` equal to 1 indicates that the decoded pictures that the annotated regions SEI message applies to are not optimized for user viewing, but rather are optimized for some other purpose such as algorithmic object classification performance. **`ar_not_optimized_for_viewing_flag`** equal to 0 indicates that the decoded pictures that the annotated regions SEI message applies to may or may not be optimized for user viewing.

`ar_true_motion_flag` equal to 1 indicates that the motion information in the coded pictures that the annotated regions SEI message applies to was selected with a goal of accurately representing object motion for objects in the annotated regions. **`ar_true_motion_flag`** equal to 0 indicates that the motion information in the coded pictures that the annotated regions SEI message applies to may or may not be selected with a goal of accurately representing object motion for objects in the annotated regions.

`ar_occluded_object_flag` equal to 1 indicates that the `ar_bounding_box_top[ar_object_idx[i]]`, `ar_bounding_box_left[ar_object_idx[i]]`, `ar_bounding_box_width[ar_object_idx[i]]`, and `ar_bounding_box_height[ar_object_idx[i]]` syntax elements represent the size and location of an object or a portion of an object that may not be visible or may be only partially visible within the cropped decoded picture. **`ar_occluded_object_flag`** equal to 0 indicates that the `ar_bounding_box_top[ar_object_idx[i]]`, `ar_bounding_box_left[ar_object_idx[i]]`, `ar_bounding_box_width[ar_object_idx[i]]`, and `ar_bounding_box_height[ar_object_idx[i]]` syntax elements represent the size and location of an object that is entirely visible within the cropped decoded picture. It is a requirement of bitstream conformance that the value of **`ar_occluded_object_flag`** shall be the same for all `annotated_regions()` syntax structures within a CLVS.

`ar_partial_object_flag_present_flag` equal to 1 indicates that `ar_partial_object_flag[ar_object_idx[i]]` syntax elements are present. **`ar_partial_object_flag_present_flag`** equal to 0 indicates that `ar_partial_object_flag[ar_object_idx[i]]` syntax elements are not present. It is a requirement of bitstream conformance that the value of **`ar_partial_object_flag_present_flag`** shall be the same for all `annotated_regions()` syntax structures within a CLVS.

`ar_object_label_present_flag` equal to 1 indicates that label information corresponding to objects in the annotated regions is present. **`ar_object_label_present_flag`** equal to 0 indicates that label information corresponding to the objects in the annotated regions is not present.

`ar_object_confidence_info_present_flag` equal to 1 indicates that `ar_object_confidence[ar_object_idx[i]]` syntax elements are present. **`ar_object_confidence_info_present_flag`** equal to 0 indicates that `ar_object_confidence[ar_object_idx[i]]` syntax elements are not present. It is a requirement of bitstream conformance that the value of **`ar_object_confidence_present_flag`** shall be the same for all `annotated_regions()` syntax structures within a CLVS.

ar_object_confidence_length_minus1 plus 1 specifies the length, in bits, of the **ar_object_confidence[ar_object_idx[i]]** syntax elements. It is a requirement of bitstream conformance that the value of **ar_object_confidence_length_minus1** shall be the same for all annotated_regions() syntax structures within a CLVS.

ar_object_label_language_present_flag equal to 1 indicates that the **ar_object_label_language** syntax element is present. **ar_object_label_language_present_flag** equal to 0 indicates that the **ar_object_label_language** syntax element is not present.

ar_bit_equal_to_zero shall be equal to zero.

ar_object_label_language contains a language tag as specified by IETF RFC 5646 followed by a null termination byte equal to 0x00. The length of the **ar_object_label_language** syntax element shall be less than or equal to 255 bytes, not including the null termination byte. When not present, the language of the label is unspecified.

ar_num_label_updates indicates the total number of labels associated with the annotated regions that are signalled. The value of **ar_num_label_updates** shall be in the range of 0 to 255, inclusive.

ar_label_idx[i] indicates the index of the signalled label. The value of **ar_label_idx[i]** shall be in the range of 0 to 255, inclusive.

ar_label_cancel_flag equal to 1 cancels the persistence scope of the **ar_label_idx[i]**-th label. **ar_label_cancel_flag** equal to 0 indicates that the **ar_label_idx[i]**-th label is assigned a signalled value.

LabelAssigned[ar_label_idx[i]] equal to 1 indicates that the **ar_label_idx[i]**-th label is assigned. **LabelAssigned[ar_label_idx[i]]** equal to 0 indicates that the **ar_label_idx[i]**-th label is not assigned.

ar_label[ar_label_idx[i]] specifies the contents of the **ar_label_idx[i]**-th label. The length of the **ar_label[ar_label_idx[i]]** syntax element shall be less than or equal to 255 bytes, not including the null termination byte.

ar_num_object_updates indicates the number of object updates to be signalled. **ar_num_object_updates** shall be in the range of 0 to 255, inclusive.

ar_object_idx[i] is the index of the object parameters to be signalled. **ar_object_idx[i]** shall be in the range of 0 to 255, inclusive.

ar_object_cancel_flag equal to 1 cancels the persistence scope of the **ar_object_idx[i]**-th object. **ar_object_cancel_flag** equal to 0 indicates that parameters associated with the **ar_object_idx[i]**-th object tracked object are signalled.

ObjectTracked[ar_object_idx[i]] equal to 1 indicates that the **object_idx[i]**-th object is tracked. **ObjectTracked[ar_object_idx[i]]** equal to 0 indicates that the **object_idx[i]**-th object is not tracked.

ar_object_label_update_flag equal to 1 indicates that an object label is signalled. **ar_object_label_update_flag** equal to 0 indicates that an object label is not signalled.

ar_object_label_idx[ar_object_idx[i]] indicates the index of the label corresponding to the **ar_object_idx[i]**-th object. When **ar_object_label_idx[ar_object_idx[i]]** is not present, its value is inferred from a previous annotated regions SEI message in output order in the same CLVS, if any. The value of **ar_object_label_idx[ar_object_idx[i]]** shall be in the range of 0 to 255, inclusive.

ar_bounding_box_update_flag equal to 1 indicates that object bounding box parameters are signalled. **ar_bounding_box_update_flag** equal to 0 indicates that object bounding box parameters are not signalled.

ar_bounding_box_cancel_flag equal to 1 cancels the persistence scope of the **ar_bounding_box_top[ar_object_idx[i]]**, **ar_bounding_box_left[ar_object_idx[i]]**, **ar_bounding_box_width[ar_object_idx[i]]**, **ar_bounding_box_height[ar_object_idx[i]]**, **ar_partial_object_flag[ar_object_idx[i]]**, and **ar_object_confidence[ar_object_idx[i]]**. **ar_bounding_box_cancel_flag** equal to 0 indicates that **ar_bounding_box_top[ar_object_idx[i]]**, **ar_bounding_box_left[ar_object_idx[i]]**, **ar_bounding_box_width[ar_object_idx[i]]**, **ar_bounding_box_height[ar_object_idx[i]]**, **ar_partial_object_flag[ar_object_idx[i]]**, and **ar_object_confidence[ar_object_idx[i]]** syntax elements are signalled.

ObjectBoundingBoxAvail[ar_object_idx[i]] equal to 1 indicates that the bounding box information of the **object_idx[i]**-th object is signalled. **ObjectBoundingBoxAvail[ar_object_idx[i]]** equal to 0 indicates that the bounding box information of the **object_idx[i]**-th object is not signalled.

ar_bounding_box_top[ar_object_idx[i]], **ar_bounding_box_left[ar_object_idx[i]]**, **ar_bounding_box_width[ar_object_idx[i]]**, and **ar_bounding_box_height[ar_object_idx[i]]** specify the coordinates of the top-left corner and the width and height, respectively, of the bounding box of the **ar_object_idx[i]**-th object in the cropped decoded picture, relative to the conformance cropping window specified by the active SPS.

Let **croppedWidth** and **croppedHeight** be the width and height, respectively, of the cropped decoded picture in units of luma samples, as specified by Equations D-28 and D-29.

The value of `ar_bounding_box_left[ar_object_idx[i]]` shall be in the range of 0 to `croppedWidth / SubWidthC - 1`, inclusive.

The value of `ar_bounding_box_top[ar_object_idx[i]]` shall be in the range of 0 to `croppedHeight / SubHeightC - 1`, inclusive.

The value of `ar_bounding_box_width[ar_object_idx[i]]` shall be in the range of 0 to `croppedWidth / SubWidthC - 1`, inclusive.

The value of `ar_bounding_box_height[ar_object_idx[i]]` shall be in the range of 0 to `croppedHeight / SubHeightC - 1`, inclusive.

The identified object rectangle contains the luma samples with horizontal picture coordinates from `SubWidthC * (conf_win_left_offset + ar_bounding_box_left[ar_object_idx[i]])` to `SubWidthC * (conf_win_left_offset + ar_bounding_box_left[ar_object_idx[i]] + ar_bounding_box_width[ar_object_idx[i]] - 1`, inclusive, and vertical picture coordinates from `SubHeightC * (conf_win_top_offset + ar_bounding_box_top[ar_object_idx[i]])` to `SubHeightC * (conf_win_top_offset + ar_bounding_box_top[ar_object_idx[i]] + ar_bounding_box_height[ar_object_idx[i]] - 1`, inclusive.

When `ChromaArrayType` is not equal to 0, the corresponding specified samples of the two chroma arrays are the samples having picture coordinates (`x / SubWidthC, y / SubHeightC`), where (`x, y`) are the picture coordinates of the specified luma samples.

The values of `ar_bounding_box_top[ar_object_idx[i]]`, `ar_bounding_box_left[ar_object_idx[i]]`, `ar_bounding_box_width[ar_object_idx[i]]` and `ar_bounding_box_height[ar_object_idx[i]]` persist in output order within the CLVS for each value of `ar_object_idx[i]`. When not present, the values of `ar_bounding_box_top[ar_object_idx[i]]`, `ar_bounding_box_left[ar_object_idx[i]]`, `ar_bounding_box_width[ar_object_idx[i]]` or `ar_bounding_box_height[ar_object_idx[i]]` are inferred from a previous annotated regions SEI message in output order in the CLVS, if any.

`ar_partial_object_flag[ar_object_idx[i]]` equal to 1 indicates that the `ar_bounding_box_top[ar_object_idx[i]]`, `ar_bounding_box_left[ar_object_idx[i]]`, `ar_bounding_box_width[ar_object_idx[i]]` and `ar_bounding_box_height[ar_object_idx[i]]` syntax elements represent the size and location of an object that is only partially visible within the cropped decoded picture. `ar_partial_object_flag[ar_object_idx[i]]` equal to 0 indicates that the `ar_bounding_box_top[ar_object_idx[i]]`, `ar_bounding_box_left[ar_object_idx[i]]`, `ar_bounding_box_width[ar_object_idx[i]]` and `ar_bounding_box_height[ar_object_idx[i]]` syntax elements represent the size and location of an object that may or may not be only partially visible within the cropped decoded picture. When not present, the value of `ar_partial_object_flag[ar_object_idx[i]]` is inferred from a previous annotated regions SEI message in output order in the CLVS, if any.

`ar_object_confidence[ar_object_idx[i]]` indicates the degree of confidence associated with the `ar_object_idx[i]`-th object, in units of $2^{-(\text{ar_object_confidence_length_minus1} + 1)}$, such that a higher value of `ar_object_confidence[ar_object_idx[i]]` indicates a higher degree of confidence. The length of the `ar_object_confidence[ar_object_idx[i]]` syntax element is `ar_object_confidence_length_minus1 + 1` bits. When not present, the value of `object_confidence[ar_object_idx[i]]` is inferred from a previous annotated regions SEI message in output order in the CLVS, if any.

D.3.48 Shutter interval information SEI message semantics

The shutter interval information SEI message indicates the shutter interval for the associated video source pictures prior to encoding, e.g., for camera-captured content, the shutter interval is amount of time that an image sensor is exposed to produce each source picture.

When a shutter interval information SEI message is present for any picture of a CLVS of a particular layer, a shutter interval information SEI message shall be present for the first picture of the CLVS. The shutter interval information SEI message persists for the current layer in decoding order from the current picture until the end of the CLVS. All shutter interval information SEI messages that apply to the same CLVS shall have the same content.

sii_time_scale specifies the number of time units that pass in one second. The value of `sii_time_scale` shall be greater than 0. For example, a time coordinate system that measures time using a 27 MHz clock has an `sii_time_scale` of 27 000 000.

fixed_shutter_interval_within_clvs_flag equal to 1 specifies that the indicated shutter interval is the same for all temporal sub-layers in the CLVS. `fixed_shutter_interval_within_clvs_flag` equal to 0 specifies that the indicated shutter interval may not be the same for all temporal sub-layers in the CLVS. When the value of `sps_max_sub_layers_minus1` is equal to 0, the value of `fixed_shutter_interval_within_clvs_flag` shall be equal to 1.

sii_num_units_in_shutter_interval, when `fixed_shutter_interval_within_clvs_flag` is equal to 1, specifies the number of time units of a clock operating at the frequency `sii_time_scale` Hz that corresponds to the indicated shutter interval of each picture in the CLVS. The value 0 may be used to indicate that the associated video content contains screen capture content, computer generated content, or other non-camera-captured content.

The indicated shutter interval, denoted by the variable `shutterInterval`, in units of seconds, is equal to the quotient of `sii_num_units_in_shutter_interval` divided by `sii_time_scale`. For example, to represent a shutter interval equal to 0.04 seconds, `sii_time_scale` may be equal to 27 000 000 and `sii_num_units_in_shutter_interval` may be equal to 1 080 000.

`sii_max_sub_layers_minus1` plus 1 specifies the maximum number of temporal sub-layers that may be present in each CLVS referring to the SPS. The value of `sii_max_sub_layers_minus1` shall be equal to the value of `sps_max_sub_layers_minus1` in the SPS.

NOTE – For example, the information conveyed in this SEI message is intended to be adequate for purposes corresponding to the use of ATSC A/341:2019 Annex D when `sii_max_sub_layers_minus1` is equal to 1 and `fixed_shutter_interval_within_clvs_flag` is equal to 0.

`sub_layer_num_units_in_shutter_interval[i]`, when present, specifies the number of time units of a clock operating at the frequency `sii_time_scale` Hz that corresponds to the shutter interval of each picture in the sub-layer representation with `TemporalId` equal to `i` in the CLVS. The sub-layer shutter interval for the sub-layer representation with `TemporalId` equal to `i`, denoted by the variable `subLayerShutterInterval[i]`, in units of seconds, is equal to the quotient of `sub_layer_num_units_in_shutter_interval[i]` divided by `sii_time_scale`.

The variable `subLayerShutterInterval[i]`, corresponding to the indicated shutter interval of each picture in the sub-layer representation with `TemporalId` equal to `i` in the CLVS, is thus derived as follows:

```

if( fixed_shutter_interval_within_clvs_flag )
    subLayerShutterInterval[ i ] = sii_num_units_in_shutter_interval ÷ sii_time_scale
else
    subLayerShutterInterval[ i ] = sub_layer_num_units_in_shutter_interval[ i ] ÷ sii_time_scale

```

(D-63)

D.3.49 Reserved SEI message semantics

The reserved SEI message consists of data reserved for future backward-compatible use by ITU-T | ISO/IEC. It is a requirement of bitstream conformance that bitstreams shall not contain reserved SEI messages until and unless the use of such messages has been specified by ITU-T | ISO/IEC. Decoders shall ignore reserved SEI messages.

Annex E

Video usability information

(This annex forms an integral part of this Recommendation | International Standard.)

E.1 General

This annex specifies syntax and semantics of the VUI parameters of the SPSs.

VUI parameters are not required for constructing the luma or chroma samples by the decoding process. Conforming decoders are not required to process this information for output order conformance to this Specification (see Annex C and clause F.13 for the specification of output order conformance). Some VUI parameters are required to check bitstream conformance and for output timing decoder conformance.

In this annex, specification for presence of VUI parameters is also satisfied when those parameters (or some subset of them) are conveyed to decoders (or to the HRD) by other means not specified in this Specification. When present in the bitstream, VUI parameters shall follow the syntax and semantics specified in this annex. When the content of VUI parameters is conveyed for the application by some means other than presence within the bitstream, the representation of the content of the VUI parameters is not required to use the same syntax specified in this annex. For the purpose of counting bits, only the appropriate bits that are actually present in the bitstream are counted.

E.2 VUI syntax

E.2.1 VUI parameters syntax

vui_parameters() {	Descriptor
aspect_ratio_info_present_flag	u(1)
if(aspect_ratio_info_present_flag) {	
aspect_ratio_idc	u(8)
if(aspect_ratio_idc == EXTENDED_SAR) {	
sar_width	u(16)
sar_height	u(16)
}	
}	
overscan_info_present_flag	u(1)
if(overscan_info_present_flag)	
overscan_appropriate_flag	u(1)
video_signal_type_present_flag	u(1)
if(video_signal_type_present_flag) {	
video_format	u(3)
video_full_range_flag	u(1)
colour_description_present_flag	u(1)
if(colour_description_present_flag) {	
colour_primaries	u(8)
transfer_characteristics	u(8)
matrix_coeffs	u(8)
}	
}	
chroma_loc_info_present_flag	u(1)
if(chroma_loc_info_present_flag) {	
chroma_sample_loc_type_top_field	ue(v)
chroma_sample_loc_type_bottom_field	ue(v)

}	
neutral_chroma_indication_flag	u(1)
field_seq_flag	u(1)
frame_field_info_present_flag	u(1)
default_display_window_flag	u(1)
if(default_display_window_flag) {	
def_disp_win_left_offset	ue(v)
def_disp_win_right_offset	ue(v)
def_disp_win_top_offset	ue(v)
def_disp_win_bottom_offset	ue(v)
}	
vui_timing_info_present_flag	u(1)
if(vui_timing_info_present_flag) {	
vui_num_units_in_tick	u(32)
vui_time_scale	u(32)
vui_poc_proportional_to_timing_flag	u(1)
if(vui_poc_proportional_to_timing_flag)	
vui_num_ticks_poc_diff_one_minus1	ue(v)
vui_hrd_parameters_present_flag	u(1)
if(vui_hrd_parameters_present_flag)	
hrd_parameters(1, sps_max_sub_layers_minus1)	
}	
bitstream_restriction_flag	u(1)
if(bitstream_restriction_flag) {	
tiles_fixed_structure_flag	u(1)
motion_vectors_over_pic_boundaries_flag	u(1)
restricted_ref_pic_lists_flag	u(1)
min_spatial_segmentation_idc	ue(v)
max_bytes_per_pic_denom	ue(v)
max_bits_per_min_cu_denom	ue(v)
log2_max_mv_length_horizontal	ue(v)
log2_max_mv_length_vertical	ue(v)
}	
}	

E.2.2 HRD parameters syntax

hrd_parameters(commonInfPresentFlag, maxNumSubLayersMinus1) {	Descriptor
if(commonInfPresentFlag) {	
nal_hrd_parameters_present_flag	u(1)
vcl_hrd_parameters_present_flag	u(1)
if(nal_hrd_parameters_present_flag vcl_hrd_parameters_present_flag) {	
sub_pic_hrd_params_present_flag	u(1)
if(sub_pic_hrd_params_present_flag) {	
tick_divisor_minus2	u(8)
du_cpb_removal_delay_increment_length_minus1	u(5)
sub_pic_cpb_params_in_pic_timing_sei_flag	u(1)
dpb_output_delay_du_length_minus1	u(5)
}	
bit_rate_scale	u(4)
cpb_size_scale	u(4)
if(sub_pic_hrd_params_present_flag)	
cpb_size_du_scale	u(4)
initial_cpb_removal_delay_length_minus1	u(5)
au_cpb_removal_delay_length_minus1	u(5)
dpb_output_delay_length_minus1	u(5)
}	
}	
for(i = 0; i <= maxNumSubLayersMinus1; i++) {	
fixed_pic_rate_general_flag[i]	u(1)
if(!fixed_pic_rate_general_flag[i])	
fixed_pic_rate_within_cvs_flag[i]	u(1)
if(fixed_pic_rate_within_cvs_flag[i])	
elemental_duration_in_tc_minus1[i]	ue(v)
else	
low_delay_hrd_flag[i]	u(1)
if(!low_delay_hrd_flag[i])	
cpb_cnt_minus1[i]	ue(v)
if(nal_hrd_parameters_present_flag)	
sub_layer_hrd_parameters(i)	
if(vcl_hrd_parameters_present_flag)	
sub_layer_hrd_parameters(i)	
}	
}	

E.2.3 Sub-layer HRD parameters syntax

sub_layer_hrd_parameters(subLayerId) {	Descriptor
for(i = 0; i < CpbCnt; i++) {	
bit_rate_value_minus1[i]	ue(v)
cpb_size_value_minus1[i]	ue(v)
if(sub_pic_hrd_params_present_flag) {	
cpb_size_du_value_minus1[i]	ue(v)
bit_rate_du_value_minus1[i]	ue(v)
}	
cbr_flag[i]	u(1)
}	
}	

E.3 VUI semantics

E.3.1 VUI parameters semantics

aspect_ratio_info_present_flag equal to 1 specifies that **aspect_ratio_idc** is present. **aspect_ratio_info_present_flag** equal to 0 specifies that **aspect_ratio_idc** is not present.

aspect_ratio_idc specifies the value of the sample aspect ratio of the luma samples. Table E.1 shows the meaning of the code. When **aspect_ratio_idc** indicates EXTENDED_SAR, the sample aspect ratio is represented by **sar_width**: **sar_height**. When the **aspect_ratio_idc** syntax element is not present, the value of **aspect_ratio_idc** is inferred to be equal to 0. Values of **aspect_ratio_idc** in the range of 17 to 254, inclusive, are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall interpret values of **aspect_ratio_idc** in the range of 17 to 254, inclusive, as equivalent to the value 0.

Table E.1 – Interpretation of sample aspect ratio indicator

aspect_ratio_idc	Sample aspect ratio	Examples of use (informative)
0	Unspecified	
1	1:1 ("square")	7680x4320 16:9 frame without horizontal overscan 3840x2160 16:9 frame without horizontal overscan 1280x720 16:9 frame without horizontal overscan 1920x1080 16:9 frame without horizontal overscan (cropped from 1920x1088) 640x480 4:3 frame without horizontal overscan
2	12:11	720x576 4:3 frame with horizontal overscan 352x288 4:3 frame without horizontal overscan
3	10:11	720x480 4:3 frame with horizontal overscan 352x240 4:3 frame without horizontal overscan
4	16:11	720x576 16:9 frame with horizontal overscan 528x576 4:3 frame without horizontal overscan
5	40:33	720x480 16:9 frame with horizontal overscan 528x480 4:3 frame without horizontal overscan
6	24:11	352x576 4:3 frame without horizontal overscan 480x576 16:9 frame with horizontal overscan
7	20:11	352x480 4:3 frame without horizontal overscan 480x480 16:9 frame with horizontal overscan
8	32:11	352x576 16:9 frame without horizontal overscan
9	80:33	352x480 16:9 frame without horizontal overscan
10	18:11	480x576 4:3 frame with horizontal overscan
11	15:11	480x480 4:3 frame with horizontal overscan
12	64:33	528x576 16:9 frame without horizontal overscan
13	160:99	528x480 16:9 frame without horizontal overscan
14	4:3	1440x1080 16:9 frame without horizontal overscan
15	3:2	1280x1080 16:9 frame without horizontal overscan
16	2:1	960x1080 16:9 frame without horizontal overscan
17..254	Reserved	
255	EXTENDED_SAR	

NOTE 1 – For the examples in Table E.1, the term "without horizontal overscan" refers to display processes in which the display area matches the area of the cropped decoded pictures and the term "with horizontal overscan" refers to display processes in which some parts near the left or right border of the cropped decoded pictures are not visible in the display area. As an example, the entry "720x576 4:3 frame with horizontal overscan" for aspect_ratio_idc equal to 2 refers to having an area of 704x576 luma samples (which has an aspect ratio of 4:3) of the cropped decoded frame (720x576 luma samples) that is visible in the display area.

NOTE 2 – For the examples in Table E.1, the frame spatial resolutions shown as examples of use would be the dimensions of the conformance cropping window when field_seq_flag is equal to 0 and would have twice the height of the dimensions of the conformance cropping window when field_seq_flag is equal to 1.

sar_width indicates the horizontal size of the sample aspect ratio (in arbitrary units).

sar_height indicates the vertical size of the sample aspect ratio (in the same arbitrary units as sar_width).

sar_width and sar_height shall be relatively prime or equal to 0. When aspect_ratio_idc is equal to 0 or sar_width is equal to 0 or sar_height is equal to 0, the sample aspect ratio is unspecified in this Specification.

overscan_info_present_flag equal to 1 specifies that the overscan_appropriate_flag is present. When overscan_info_present_flag is equal to 0 or is not present, the preferred display method for the video signal is unspecified.

overscan_appropriate_flag equal to 1 indicates that the cropped decoded pictures output are suitable for display using overscan. overscan_appropriate_flag equal to 0 indicates that the cropped decoded pictures output contain visually important information in the entire region out to the edges of the conformance cropping window of the picture, such that the cropped decoded pictures output should not be displayed using overscan. Instead, they should be displayed using either an exact match between the display area and the conformance cropping window, or using underscan. As used in this paragraph, the term "overscan" refers to display processes in which some parts near the borders of the cropped decoded

pictures are not visible in the display area. The term "underscan" describes display processes in which the entire cropped decoded pictures are visible in the display area, but they do not cover the entire display area. For display processes that neither use overscan nor underscan, the display area exactly matches the area of the cropped decoded pictures.

NOTE 3 – For example, `overscan_appropriate_flag` equal to 1 might be used for entertainment television programming, or for a live view of people in a videoconference and `overscan_appropriate_flag` equal to 0 might be used for computer screen capture or security camera content.

`video_signal_type_present_flag` equal to 1 specifies that `video_format`, `video_full_range_flag` and `colour_description_present_flag` are present. `video_signal_type_present_flag` equal to 0, specify that `video_format`, `video_full_range_flag` and `colour_description_present_flag` are not present.

NOTE 4 – Some of the semantics of video signal type parameters associated with `video_signal_type_present_flag` equal to 1 are expressed in terms of the properties of source pictures prior to operation of the encoding process, which is outside the scope of this Specification. This is partly for historical reasons and due to the common general practice of how the indicated data is typically described in industry publications. The actual intent for providing this syntax in the bitstream is to assist decoding systems to properly interpret and make effective use of the decoded video pictures, e.g., for use by the display process (which is also outside the scope of this Specification, but for which having an indication of how the pictures should be interpreted is important).

`video_format` indicates the representation of the pictures as specified in Table E.2, before being coded in accordance with this Specification. When the `video_format` syntax element is not present, the value of `video_format` is inferred to be equal to 5. The values 6 and 7 for `video_format` are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall interpret the values 6 and 7 for `video_format` as equivalent to the value 5.

Table E.2 – Meaning of `video_format`

<code>video_format</code>	Meaning
0	Component
1	PAL
2	NTSC
3	SECAM
4	MAC
5	Unspecified video format

`video_full_range_flag` indicates the black level and range of the luma and chroma signals as derived from E'_Y , E'_{PB} , and E'_{PR} or E'_R , E'_G , and E'_B real-valued component signals.

When the `video_full_range_flag` syntax element is not present, the value of `video_full_range_flag` is inferred to be equal to 0.

`colour_description_present_flag` equal to 1 specifies that `colour_primaries`, `transfer_characteristics`, and `matrix_coeffs` are present. `colour_description_present_flag` equal to 0 specifies that `colour_primaries`, `transfer_characteristics`, and `matrix_coeffs` are not present.

`colour_primaries` indicates the chromaticity coordinates of the source primaries as specified in Table E.3 in terms of the CIE 1931 definition of x and y as specified in ISO 11664-1.

When the `colour_primaries` syntax element is not present, the value of `colour_primaries` is inferred to be equal to 2 (the chromaticity is unspecified or is determined by the application). Values of `colour_primaries` that are identified as reserved in Table E.3 are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall interpret reserved values of `colour_primaries` as equivalent to the value 2.

Table E.3 – Colour primaries interpretation using the colour_primaries syntax element

Value	Primaries			Informative remark
0	Reserved			For future use by ITU-T ISO/IEC
1	primary	x	y	Rec. ITU-R BT.709-6 Rec. ITU-R BT.1361-0 conventional colour gamut system and extended colour gamut system (historical) IEC 61966-2-1 sRGB or sYCC IEC 61966-2-4 SMPTE RP 177 (1993) Annex B
	green	0.300	0.600	
	blue	0.150	0.060	
	red	0.640	0.330	
	white D65	0.3127	0.3290	
2	Unspecified			Image characteristics are unknown or are determined by the application.
3	Reserved			For future use by ITU-T ISO/IEC
4	primary	x	y	Rec. ITU-R BT.470-6 System M (historical) NTSC Recommendation for transmission standards for colour television (1953) FCC Title 47 Code of Federal Regulations (2003) 73.682 (a) (20)
	green	0.21	0.71	
	blue	0.14	0.08	
	red	0.67	0.33	
	white C	0.310	0.316	
5	primary	x	y	Rec. ITU-R BT.470-6 System B, G (historical) Rec. ITU-R BT.601-7 625 Rec. ITU-R BT.1358-0 625 (historical) Rec. ITU-R BT.1700-0 625 PAL and 625 SECAM
	green	0.29	0.60	
	blue	0.15	0.06	
	red	0.64	0.33	
	white D65	0.3127	0.3290	
6	primary	x	y	Rec. ITU-R BT.601-7 525 Rec. ITU-R BT.1358-1 525 or 625 (historical) Rec. ITU-R BT.1700-0 NTSC SMPTE ST 170 (2004) (functionally the same as the value 7)
	green	0.310	0.595	
	blue	0.155	0.070	
	red	0.630	0.340	
	white D65	0.3127	0.3290	
7	primary	x	y	SMPTE ST 240 (1999, historical) (functionally the same as the value 6)
	green	0.310	0.595	
	blue	0.155	0.070	
	red	0.630	0.340	
	white D65	0.3127	0.3290	
8	primary	x	y	Generic film (colour filters using Illuminant C)
	green	0.243	0.692 (Wratten 58)	
	blue	0.145	0.049 (Wratten 47)	
	red	0.681	0.319 (Wratten 25)	
	white C	0.310	0.316	
9	primary	x	y	Rec. ITU-R BT.2020-2 Rec. ITU-R BT.2100-2
	green	0.170	0.797	
	blue	0.131	0.046	
	red	0.708	0.292	
	white D65	0.3127	0.3290	
10	primary	x	y	SMPTE ST 428-1 (2006) (CIE 1931 XYZ)
	green (Y)	0.0	1.0	
	blue (Z)	0.0	0.0	
	red (X)	1.0	0.0	
	centre white	1 ÷ 3	1 ÷ 3	
11	primary	x	y	SMPTE RP 431-2 (2011) SMPTE ST 2113 (2019) "P3DCI"
	green	0.265	0.690	
	blue	0.150	0.060	
	red	0.680	0.320	
	white	0.314	0.351	

Table E.3 – Colour primaries interpretation using the colour_primaries syntax element

Value	Primaries			Informative remark
12	primary	x	y	SMPTE EG 432-1 (2010) SMPTE ST 2113 (2019) "P3D65"
	green	0.265	0.690	
	blue	0.150	0.060	
	red	0.680	0.320	
	white D65	0.3127	0.3290	
13..21	Reserved			For future use by ITU-T ISO/IEC
22	primary	x	y	EBU Tech. 3213-E (1975)
	green	0.295	0.605	
	blue	0.155	0.077	
	red	0.630	0.340	
	white D65	0.3127	0.3290	
23..255	Reserved			For future use by ITU-T ISO/IEC

transfer_characteristics, as specified in Table E.4, either indicates the reference opto-electronic transfer characteristic function of the source picture as a function of a source input linear optical intensity L_c with a nominal real-valued range of 0 to 1 or indicates the inverse of the reference electro-optical transfer characteristic function as a function of an output linear optical intensity L_o with a nominal real-valued range of 0 to 1. For interpretation of entries in Table E.4 that are expressed in terms of multiple curve segments parameterized by the variable α over a region bounded by the variable β or by the variables β and γ , the values of α and β are defined to be the positive constants necessary for the curve segments that meet at the value β to have continuity of value and continuity of slope at the value β , and the value of γ , when applicable, is defined to be the positive constant necessary for the associated curve segments to meet at the value γ . For example, for transfer_characteristics equal to 1, 6, 11, 14, or 15, α has the value $1 + 5.5 * \beta = 1.099\ 296\ 826\ 809\ 442\dots$ and β has the value 0.018 053 968 510 807....

When the transfer_characteristics syntax element is not present, the value of transfer_characteristics is inferred to be equal to 2 (the transfer characteristics are unspecified or are determined by the application). Values of transfer_characteristics that are identified as reserved in Table E.4 are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall interpret reserved values of transfer_characteristics as equivalent to the value 2.

NOTE 5 – As indicated in Table E.4, some values of transfer_characteristics are defined in terms of a reference opto-electronic transfer characteristic function and others are defined in terms of a reference electro-optical transfer characteristic function, according to the convention that has been applied in other Specifications. In the cases of Rec. ITU-R BT.709-6 and Rec. ITU-R BT.2020-2 (which may be indicated by transfer_characteristics equal to 1, 6, 14, or 15), although the value is defined in terms of a reference opto-electronic transfer characteristic function, a suggested corresponding reference electro-optical transfer characteristic function for flat panel displays used in HDTV studio production has been specified in Rec. ITU-R BT.1886-0.

Table E.4 – Transfer characteristics interpretation using the transfer_characteristics syntax element

Value	Transfer characteristic	Informative remark
0	Reserved	For future use by ITU-T ISO/IEC
1	$V = \alpha * L_c^{0.45} - (\alpha - 1)$ for $1 \geq L_c \geq \beta$ $V = 4.500 * L_c$ for $\beta > L_c \geq 0$	Rec. ITU-R BT.709-6 Rec. ITU-R BT.1361-0 conventional colour gamut system (historical) (functionally the same as the values 6, 14, and 15)
2	Unspecified	Image characteristics are unknown or are determined by the application.
3	Reserved	For future use by ITU-T ISO/IEC
4	Assumed display gamma 2.2	Rec. ITU-R BT.470-6 System M (historical) NTSC Recommendation for transmission standards for colour television (1953) FCC, Title 47 Code of Federal Regulations (2003) 73.682 (a) (20)

Table E.4 – Transfer characteristics interpretation using the transfer_characteristics syntax element

Value	Transfer characteristic	Informative remark
5	Assumed display gamma 2.8	Rec. ITU-R BT.470-6 System B, G (historical) Rec. ITU-R BT.1700-0 625 PAL and 625 SECAM
6	$V = \alpha * L_c^{0.45} - (\alpha - 1)$ for $1 \geq L_c \geq \beta$ $V = 4.500 * L_c$ for $\beta > L_c \geq 0$	Rec. ITU-R BT.601-7 525 or 625 Rec. ITU-R BT.1358-1 525 or 625 (historical) Rec. ITU-R BT.1700-0 NTSC SMPTE ST 170 (2004) (functionally the same as the values 1, 14, and 15)
7	$V = \alpha * L_c^{0.45} - (\alpha - 1)$ for $1 \geq L_c \geq \beta$ $V = 4.0 * L_c$ for $\beta > L_c \geq 0$	SMPTE ST 240 (1999, historical)
8	$V = L_c$ for all values of L_c	Linear transfer characteristics
9	$V = 1.0 + \text{Log}10(L_c) \div 2$ for $1 \geq L_c \geq 0.01$ $V = 0.0$ for $0.01 > L_c \geq 0$	Logarithmic transfer characteristic (100:1 range)
10	$V = 1.0 + \text{Log}10(L_c) \div 2.5$ for $1 \geq L_c \geq \text{Sqrt}(10) \div 1\,000$ $V = 0.0$ for $\text{Sqrt}(10) \div 1\,000 > L_c \geq 0$	Logarithmic transfer characteristic (100 * Sqrt(10) : 1 range)
11	$V = \alpha * L_c^{0.45} - (\alpha - 1)$ for $L_c \geq \beta$ $V = 4.500 * L_c$ for $\beta > L_c > -\beta$ $V = -\alpha * (-L_c)^{0.45} + (\alpha - 1)$ for $-\beta \geq L_c$	IEC 61966-2-4
12	$V = \alpha * L_c^{0.45} - (\alpha - 1)$ for $1.33 > L_c \geq \beta$ $V = 4.500 * L_c$ for $\beta > L_c \geq -\gamma$ $V = -(\alpha * (-4 * L_c)^{0.45} - (\alpha - 1)) \div 4$ for $-\gamma > L_c \geq -0.25$	Rec. ITU-R BT.1361-0 extended colour gamut system (historical)
13	– If matrix_coeffs is equal to 0 $V = \alpha * L_c^{(1 \div 2.4)} - (\alpha - 1)$ for $1 \geq L_c \geq \beta$ $V = 12.92 * L_c$ for $\beta > L_c \geq 0$ – Otherwise $V = \alpha * L_c^{(1 \div 2.4)} - (\alpha - 1)$ for $L_c \geq \beta$ $V = 12.92 * L_c$ for $\beta > L_c > -\beta$ $V = -\alpha * (-L_c)^{(1 \div 2.4)} + (\alpha - 1)$ for $-\beta \geq L_c$	IEC 61966-2-1 sRGB (with matrix_coeffs equal to 0) IEC 61966-2-1 sYCC (with matrix_coeffs equal to 5)
14	$V = \alpha * L_c^{0.45} - (\alpha - 1)$ for $1 \geq L_c \geq \beta$ $V = 4.500 * L_c$ for $\beta > L_c \geq 0$	Rec. ITU-R BT.2020-2 (functionally the same as the values 1, 6, and 15)
15	$V = \alpha * L_c^{0.45} - (\alpha - 1)$ for $1 \geq L_c \geq \beta$ $V = 4.500 * L_c$ for $\beta > L_c \geq 0$	Rec. ITU-R BT.2020-2 (functionally the same as the values 1, 6, and 14)
16	$V = ((c_1 + c_2 * L_o^n) \div (1 + c_3 * L_o^n))^m$ for all values of L_o $c_1 = c_3 - c_2 + 1 = 3\,424 \div 4\,096 = 0.835\,937\,5$ $c_2 = 32 * 2\,413 \div 4\,096 = 18.851\,562\,5$ $c_3 = 32 * 2\,392 \div 4\,096 = 18.687\,5$ $m = 128 * 2\,523 \div 4\,096 = 78.843\,75$ $n = 0.25 * 2\,610 \div 4\,096 = 0.159\,301\,757\,812\,5$ for which L_o equal to 1 for peak white is ordinarily intended to correspond to a reference output luminance level of 10 000 candelas per square metre	SMPTE ST 2084 (2014) for 10, 12, 14, and 16-bit systems Rec. ITU-R BT.2100-2 perceptual quantization (PQ) system
17	$V = (48 * L_o \div 52.37)^{(1 \div 2.6)}$ for all values of L_o for which L_o equal to 1 for peak white is ordinarily intended to correspond to a reference output luminance level of 48 candelas per square metre	SMPTE ST 428-1 (2006)
18	$V = a * \text{Ln}(12 * L_c - b) + c$ for $1 \geq L_c > 1 \div 12$ $V = \text{Sqrt}(3) * L_c^{0.5}$ for $1 \div 12 \geq L_c \geq 0$ $a = 0.178\,832\,77$, $b = 0.284\,668\,92$, $c = 0.559\,910\,73$	Association of Radio Industries and Businesses (ARIB) STD-B67 Rec. ITU-R BT.2100-2 hybrid log-gamma (HLG) system
19..255	Reserved	For future use by ITU-T ISO/IEC

NOTE 6 – For transfer_characteristics equal to 18, the equations given in Table E.4 are normalized for a source input linear optical intensity L_c with a nominal real-valued range of 0 to 1. An alternative scaling that is mathematically equivalent is used in ARIB STD-B67 with the source input linear optical intensity having a nominal real-valued range of 0 to 12.

matrix_coeffs describes the matrix coefficients used in deriving luma and chroma signals from the green, blue, and red, or Y, Z, and X primaries, as specified in Table E.5.

matrix_coeffs shall not be equal to 0 unless both of the following conditions are true:

- BitDepth_C is equal to BitDepth_Y.
- chroma_format_idc is equal to 3 (the 4:4:4 chroma format).

The specification of the use of matrix_coeffs equal to 0 under all other conditions is reserved for future use by ITU-T | ISO/IEC.

matrix_coeffs shall not be equal to 8 unless one of the following conditions is true:

- BitDepth_C is equal to BitDepth_Y,
- BitDepth_C is equal to BitDepth_Y + 1 and chroma_format_idc is equal to 3 (the 4:4:4 chroma format).

The specification of the use of matrix_coeffs equal to 8 under all other conditions is reserved for future use by ITU-T | ISO/IEC.

When the matrix_coeffs syntax element is not present, the value of matrix_coeffs is inferred to be equal to 2 (unspecified).

The interpretation of matrix_coeffs, together with colour_primaries and transfer_characteristics, is specified by the equations below.

NOTE 7 – For purposes of YZX representation when matrix_coeffs is equal to 0, the symbols R, G, and B are substituted for X, Y, and Z, respectively, in the following descriptions of Equations E-1 to E-3, E-13 to E-15, E-19 to E-21, and E-31 to E-33.

E_R , E_G , and E_B are defined as "linear-domain" real-valued signals based on the indicated colour primaries before application of the transfer characteristics function.

Nominal peak white is specified as having E_R equal to 1, E_G equal to 1, and E_B equal to 1.

Nominal black is specified as having E_R equal to 0, E_G equal to 0, and E_B equal to 0.

The application of the transfer characteristics function is denoted by $(x)'$ for an argument x .

- If matrix_coeffs is not equal to 14, the signals E'_R , E'_G , and E'_B are determined by application of the transfer characteristics function as follows:

$$E'_R = (E_R)' \quad (E-1)$$

$$E'_G = (E_G)' \quad (E-2)$$

$$E'_B = (E_B)' \quad (E-3)$$

In this case, the range of E'_R , E'_G , and E'_B is specified as follows:

- If transfer_characteristics is equal to 11 or 12, or transfer_characteristics is equal to 13 and matrix_coeffs is not equal to 0, E'_R , E'_G , and E'_B are real numbers with values that have a larger range than the range of 0 to 1, inclusive, and their range is not specified in this Specification.
- Otherwise, E'_R , E'_G and E'_B are real numbers in the range of 0 to 1.
- Otherwise (matrix_coeffs is equal to 14), the "linear-domain" real-valued signals E_L , E_M , and E_S are determined as follows:

$$E_L = (1\,688 * E_R + 2\,146 * E_G + 262 * E_B) \div 4\,096 \quad (E-4)$$

$$E_M = (683 * E_R + 2\,951 * E_G + 462 * E_B) \div 4\,096 \quad (E-5)$$

$$E_S = (99 * E_R + 309 * E_G + 3\,688 * E_B) \div 4\,096 \quad (E-6)$$

In this case, the signals E'_L , E'_M , and E'_S are determined by application of the transfer characteristics function as follows:

$$E'_L = (E_L)' \quad (E-7)$$

$$E'_M = (E_M)' \quad (E-8)$$

$$E'_S = (E_S)' \quad (E-9)$$

The interpretation of `matrix_coeffs` is specified as follows:

- If `video_full_range_flag` is equal to 0, the following applies:

- If `matrix_coeffs` is equal to 1, 4, 5, 6, 7, 9, 10, 11, 12, 13, or 14, the following equations apply:

$$Y = \text{Clip1}_Y(\text{Round}((1 \ll (\text{BitDepth}_Y - 8)) * (219 * E'_Y + 16))) \quad (E-10)$$

$$Cb = \text{Clip1}_C(\text{Round}((1 \ll (\text{BitDepth}_C - 8)) * (224 * E'_{PB} + 128))) \quad (E-11)$$

$$Cr = \text{Clip1}_C(\text{Round}((1 \ll (\text{BitDepth}_C - 8)) * (224 * E'_{PR} + 128))) \quad (E-12)$$

- Otherwise, if `matrix_coeffs` is equal to 0 or 8, the following equations apply:

$$R = \text{Clip1}_Y((1 \ll (\text{BitDepth}_Y - 8)) * (219 * E'_R + 16)) \quad (E-13)$$

$$G = \text{Clip1}_Y((1 \ll (\text{BitDepth}_Y - 8)) * (219 * E'_G + 16)) \quad (E-14)$$

$$B = \text{Clip1}_Y((1 \ll (\text{BitDepth}_Y - 8)) * (219 * E'_B + 16)) \quad (E-15)$$

- Otherwise, if `matrix_coeffs` is equal to 2, the interpretation of the `matrix_coeffs` syntax element is unknown or is determined by the application.
- Otherwise (`matrix_coeffs` is not equal to 0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, or 14), the interpretation of the `matrix_coeffs` syntax element is reserved for future definition by ITU-T | ISO/IEC.

- Otherwise (`video_full_range_flag` is equal to 1), the following applies:

- If `matrix_coeffs` is equal to 1, 4, 5, 6, 7, 9, 10, 11, 12, 13, or 14, the following applies:

$$Y = \text{Clip1}_Y(\text{Round}(((1 \ll \text{BitDepth}_Y) - 1) * E'_Y)) \quad (E-16)$$

$$Cb = \text{Clip1}_C(\text{Round}(((1 \ll \text{BitDepth}_C) - 1) * E'_{PB} + (1 \ll (\text{BitDepth}_C - 1)))) \quad (E-17)$$

$$Cr = \text{Clip1}_C(\text{Round}(((1 \ll \text{BitDepth}_C) - 1) * E'_{PR} + (1 \ll (\text{BitDepth}_C - 1)))) \quad (E-18)$$

- Otherwise, if `matrix_coeffs` is equal to 0 or 8, the following applies:

$$R = \text{Clip1}_Y(((1 \ll \text{BitDepth}_Y) - 1) * E'_R) \quad (E-19)$$

$$G = \text{Clip1}_Y(((1 \ll \text{BitDepth}_Y) - 1) * E'_G) \quad (E-20)$$

$$B = \text{Clip1}_Y(((1 \ll \text{BitDepth}_Y) - 1) * E'_B) \quad (E-21)$$

- Otherwise, if `matrix_coeffs` is equal to 2, the interpretation of the `matrix_coeffs` syntax element is unknown or is determined by the application.
- Otherwise (`matrix_coeffs` is not equal to 0, 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, or 14), the interpretation of the `matrix_coeffs` syntax element is reserved for future definition by ITU-T | ISO/IEC. Reserved values for `matrix_coeffs` shall not be present in bitstreams conforming to this version of this Specification. Decoders shall interpret reserved values of `matrix_coeffs` as equivalent to the value 2.

It is a requirement of bitstream conformance to this version of this Specification that when `colour_primaries` is not equal to 1, 4, 5, 6, 7, 8, 9, 10, 11, 12, or 22, `matrix_coeffs` shall not be equal to 12 or 13.

When `matrix_coeffs` is equal to 1, 4, 5, 6, 7, 9, 10, 11, 12, or 13, the constants K_B and K_R are specified as follows:

- If `matrix_coeffs` is not equal to 12 or 13, the constants K_B and K_R are specified in Table E.5.

- Otherwise (matrix_coeffs is equal to 12 or 13), the constants K_R and K_B are computed as follows, using the chromaticity coordinates (x_R, y_R) , (x_G, y_G) , (x_B, y_B) , and (x_W, y_W) specified by Table E.3 for the colour primaries syntax element for the red, green, blue, and white colour primaries, respectively.

$$K_R = \frac{y_R * (x_W * (y_G * z_B - y_B * z_G) + y_W * (x_B * z_G - x_G * z_B) + z_W * (x_G * y_B - x_B * y_G))}{y_W * (x_R * (y_G * z_B - y_B * z_G) + x_G * (y_B * z_R - y_R * z_B) + x_B * (y_R * z_G - y_G * z_R))} \quad (E-22)$$

$$K_B = \frac{y_B * (x_W * (y_R * z_G - y_G * z_R) + y_W * (x_G * z_R - x_R * z_G) + z_W * (x_R * y_G - x_G * y_R))}{y_W * (x_R * (y_G * z_B - y_B * z_G) + x_G * (y_B * z_R - y_R * z_B) + x_B * (y_R * z_G - y_G * z_R))} \quad (E-23)$$

where the values of z_R , z_G , z_B , and z_W , are given by.

$$z_R = 1 - (x_R + y_R) \quad (E-24)$$

$$z_G = 1 - (x_G + y_G) \quad (E-25)$$

$$z_B = 1 - (x_B + y_B) \quad (E-26)$$

$$z_W = 1 - (x_W + y_W) \quad (E-27)$$

The variables E'_Y , E'_{PB} , and E'_{PR} (for matrix_coeffs not equal to 0 or 8) or Y , C_b , and C_r (for matrix_coeffs equal to 0 or 8) are specified as follows:

- If matrix_coeffs is not equal to 0, 8, 10, 11, 13, or 14, the following equations apply:

$$E'_Y = K_R * E'_R + (1 - K_R - K_B) * E'_G + K_B * E'_B \quad (E-28)$$

$$E'_{PB} = 0.5 * (E'_B - E'_Y) \div (1 - K_B) \quad (E-29)$$

$$E'_{PR} = 0.5 * (E'_R - E'_Y) \div (1 - K_R) \quad (E-30)$$

NOTE 8 – E'_Y is a real number with the value 0 associated with nominal black and the value 1 associated with nominal white. E'_{PB} and E'_{PR} are real numbers with the value 0 associated with both nominal black and nominal white. When transfer_characteristics is not equal to 11 or 12, E'_Y is a real number with values in the range of 0 to 1 inclusive. When transfer_characteristics is not equal to 11 or 12, E'_{PB} and E'_{PR} are real numbers with values in the range of -0.5 to 0.5 inclusive. When transfer_characteristics is equal to 11 or 12, E'_Y , E'_{PB} , and E'_{PR} are real numbers with a larger range not specified in this Specification.

- Otherwise, if matrix_coeffs is equal to 0, the following equations apply:

$$Y = \text{Round}(G) \quad (E-31)$$

$$C_b = \text{Round}(B) \quad (E-32)$$

$$C_r = \text{Round}(R) \quad (E-33)$$

- Otherwise, if matrix_coeffs is equal to 8, the following applies:

- If BitDepth_C is equal to BitDepth_Y, the following equations apply:

$$Y = \text{Round}(0.5 * G + 0.25 * (R + B)) \quad (E-34)$$

$$C_b = \text{Round}(0.5 * G - 0.25 * (R + B)) + (1 \ll (\text{BitDepth}_C - 1)) \quad (E-35)$$

$$C_r = \text{Round}(0.5 * (R - B)) + (1 \ll (\text{BitDepth}_C - 1)) \quad (E-36)$$

NOTE 9 – In this case, for purposes of the YCgCo nomenclature used in Table E.5, C_b and C_r of Equations E-35 and E-36 may be referred to as C_g and C_o , respectively. An appropriate inverse conversion for Equations E-34 to E-36 is as follows:

$$t = Y - (C_b - (1 \ll (\text{BitDepth}_C - 1))) \quad (E-37)$$

$$G = \text{Clip1}_Y(Y + (C_b - (1 \ll (\text{BitDepth}_C - 1)))) \quad (E-38)$$

$$B = \text{Clip1}_Y(t - (C_r - (1 \ll (\text{BitDepth}_C - 1)))) \quad (E-39)$$

$$R = \text{Clip1}_Y(t + (C_r - (1 \ll (\text{BitDepth}_C - 1)))) \quad (E-40)$$

- Otherwise (BitDepth_C is not equal to BitDepth_Y), the following equations apply:

$$Cr = \text{Round}(R) - \text{Round}(B) + (1 \ll (\text{BitDepth}_C - 1)) \quad (\text{E-41})$$

$$t = \text{Round}(B) + ((Cr - (1 \ll (\text{BitDepth}_C - 1))) \gg 1) \quad (\text{E-42})$$

$$Cb = \text{Round}(G) - t + (1 \ll (\text{BitDepth}_C - 1)) \quad (\text{E-43})$$

$$Y = t + ((Cb - (1 \ll (\text{BitDepth}_C - 1))) \gg 1) \quad (\text{E-44})$$

NOTE 10 – In this case, for purposes of the YCgCo nomenclature used in Table E.5, Cb and Cr of Equations E-43 and E-41 may be referred to as Cg and Co, respectively. An appropriate inverse conversion for Equations E-41 to E-44 is as follows:

$$t = Y - ((Cb - (1 \ll (\text{BitDepth}_C - 1))) \gg 1) \quad (\text{E-45})$$

$$G = \text{Clip1}_Y(t + (Cb - (1 \ll (\text{BitDepth}_C - 1)))) \quad (\text{E-46})$$

$$B = \text{Clip1}_Y(t - ((Cr - (1 \ll (\text{BitDepth}_C - 1))) \gg 1)) \quad (\text{E-47})$$

$$R = \text{Clip1}_Y(B + (Cr - (1 \ll (\text{BitDepth}_C - 1)))) \quad (\text{E-48})$$

- Otherwise, if matrix_coeffs is equal to 10 or 13, the signal E'_Y is determined by application of the transfer characteristics function as follows and Equations E-51 to E-54 apply for specification of the signals E'_{PB} and E'_{PR} :

$$E_Y = K_R * E_R + (1 - K_R - K_B) * E_G + K_B * E_B \quad (\text{E-49})$$

$$E'_Y = (E_Y)' \quad (\text{E-50})$$

NOTE 11 – In this case, E_Y is defined from the "linear-domain" signals for E_R , E_G , and E_B , prior to application of the transfer characteristics function, which is then applied to produce the signal E'_Y . E_Y and E'_Y are real values with the value 0 associated with nominal black and the value 1 associated with nominal white.

while the signals E'_{PB} and E'_{PR} are determined as follows:

$$E'_{PB} = (E'_B - E'_Y) \div (2 * N_B) \quad \text{for } -N_B \leq E'_B - E'_Y \leq 0 \quad (\text{E-51})$$

$$E'_{PB} = (E'_B - E'_Y) \div (2 * P_B) \quad \text{for } 0 < E'_B - E'_Y \leq P_B \quad (\text{E-52})$$

$$E'_{PR} = (E'_R - E'_Y) \div (2 * N_R) \quad \text{for } -N_R \leq E'_R - E'_Y \leq 0 \quad (\text{E-53})$$

$$E'_{PR} = (E'_R - E'_Y) \div (2 * P_R) \quad \text{for } 0 < E'_R - E'_Y \leq P_R \quad (\text{E-54})$$

where the constants N_B , P_B , N_R , and P_R are determined by application of the transfer characteristics function to expressions involving the constants K_B and K_R as follows:

$$N_B = (1 - K_B)' \quad (\text{E-55})$$

$$P_B = 1 - (K_B)' \quad (\text{E-56})$$

$$N_R = (1 - K_R)' \quad (\text{E-57})$$

$$P_R = 1 - (K_R)' \quad (\text{E-58})$$

- Otherwise, if matrix_coeffs is equal to 11, the following equations apply:

$$E'_Y = E'_G \quad (\text{E-59})$$

$$E'_{PB} = 0.5 * (0.986\,566 * E'_B - E'_Y) \quad (\text{E-60})$$

$$E'_{PR} = 0.5 * (E'_R - 0.991\,902 * E'_Y) \quad (E-61)$$

NOTE 12 – In this case, for purposes of the Y'D'zD'x nomenclature used in Table E.5, E'PB may be referred to as D'z and E'PR may be referred to as D'x.

- Otherwise (matrix_coefs is equal to 14), the following equations apply:
 - If transfer_characteristics is not equal to 18, the following equations apply:

$$E'_Y = 0.5 * (E'_L + E'_M) \quad (E-62)$$

$$E'_{PB} = (6\,610 * E'_L - 13\,613 * E'_M + 7\,003 * E'_S) \div 4\,096 \quad (E-63)$$

$$E'_{PR} = (17\,933 * E'_L - 17\,390 * E'_M - 543 * E'_S) \div 4\,096 \quad (E-64)$$

- Otherwise, the following equations apply:

$$E'_Y = 0.5 * (E'_L + E'_M) \quad (E-65)$$

$$E'_{PB} = (3\,625 * E'_L - 7\,465 * E'_M + 3\,840 * E'_S) \div 4\,096 \quad (E-66)$$

$$E'_{PR} = (9\,500 * E'_L - 9\,212 * E'_M - 288 * E'_S) \div 4\,096 \quad (E-67)$$

NOTE 13 – In this case, for purposes of the IC_TC_P nomenclature used in Table E.5, E'_Y, E'_{PB}, and E'_{PR} of Equations E-62, E-63, and E-64 or Equations E-65, E-66, and E-67 may be referred to as I, C_T, and C_P, respectively. Equations E-62 to E-64 were designed specifically for use with transfer_characteristics equal to 16 (PQ), and Equations E-65 to E-67 were designed specifically for use with transfer_characteristics equal to 18 (HLG).

Table E.5 – Matrix coefficients interpretation using the matrix_coeffs syntax element

Value	Matrix	Informative remark
0	Identity	The identity matrix. Typically used for GBR (often referred to as RGB); however, may also be used for YZX (often referred to as XYZ) IEC 61966-2-1 sRGB SMPTE ST 428-1 (2006) See Equations E-31 to E-33
1	$K_R = 0.2126; K_B = 0.0722$	Rec. ITU-R BT.709-6 Rec. ITU-R BT.1361-0 conventional colour gamut system and extended colour gamut system (historical) IEC 61966-2-4 xvYCC ₇₀₉ SMPTE RP 177 (1993) Annex B See Equations E-28 to E-30
2	Unspecified	Image characteristics are unknown or are determined by the application.
3	Reserved	For future use by ITU-T ISO/IEC
4	$K_R = 0.30; K_B = 0.11$	FCC Title 47 Code of Federal Regulations (2003) 73.682 (a) (20) See Equations E-28 to E-30
5	$K_R = 0.299; K_B = 0.114$	Rec. ITU-R BT.470-6 System B, G (historical) Rec. ITU-R BT.601-7 625 Rec. ITU-R BT.1358-0 625 (historical) Rec. ITU-R BT.1700-0 625 PAL and 625 SECAM IEC 61966-2-1 sYCC IEC 61966-2-4 xvYCC ₆₀₁ (functionally the same as the value 6) See Equations E-28 to E-30
6	$K_R = 0.299; K_B = 0.114$	Rec. ITU-R BT.601-7 525 Rec. ITU-R BT.1358-1 525 or 625 (historical) Rec. ITU-R BT.1700-0 NTSC SMPTE ST 170 (2004) (functionally the same as the value 5) See Equations E-28 to E-30
7	$K_R = 0.212; K_B = 0.087$	SMPTE ST 240 (1999, historical) See Equations E-28 to E-30
8	YCgCo	See Equations E-34 to E-48
9	$K_R = 0.2627; K_B = 0.0593$	Rec. ITU-R BT.2020-2 non-constant luminance system Rec. ITU-R BT.2100-2 Y'CbCr See Equations E-28 to E-30
10	$K_R = 0.2627; K_B = 0.0593$	Rec. ITU-R BT.2020-2 constant luminance system See Equations E-49 to E-58
11	$Y'D'_zD'_x$	SMPTE ST 2085 (2015) See Equations E-59 to E-61
12	See Equations E-22 to E-27	Chromaticity-derived non-constant luminance system See Equations E-28 to E-30
13	See Equations E-22 to E-27	Chromaticity-derived constant luminance system See Equations E-49 to E-58
14	$IC_T C_P$	Rec. ITU-R BT.2100-2 $IC_T C_P$ See Equations E-62 to E-64 for transfer_characteristics value 16 (PQ) See Equations E-65 to E-67 for transfer_characteristics value 18 (HLG)
15..255	Reserved	For future use by ITU-T ISO/IEC

chroma_loc_info_present_flag equal to 1 specifies that **chroma_sample_loc_type_top_field** and **chroma_sample_loc_type_bottom_field** are present. **chroma_loc_info_present_flag** equal to 0 specifies that **chroma_sample_loc_type_top_field** and **chroma_sample_loc_type_bottom_field** are not present.

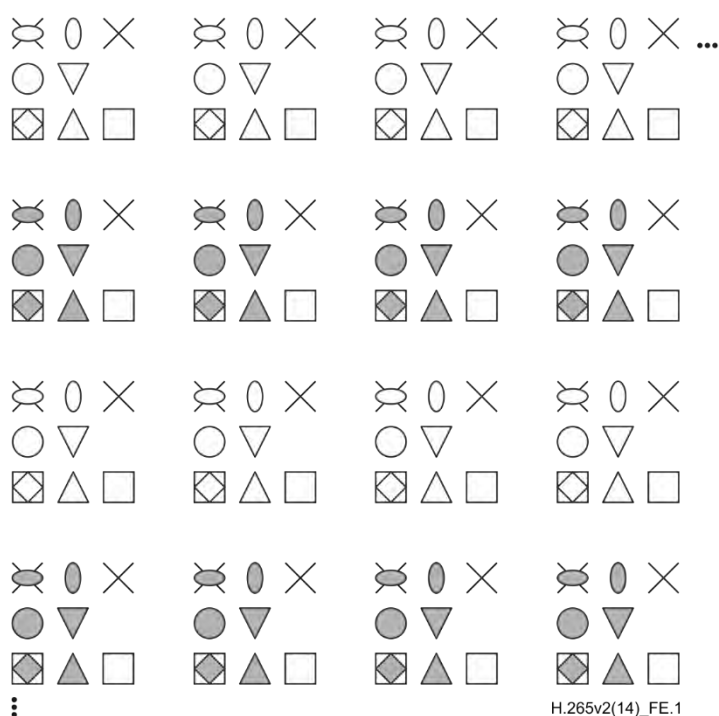
When **chroma_format_idc** is not equal to 1, **chroma_loc_info_present_flag** should be equal to 0.

chroma_sample_loc_type_top_field and **chroma_sample_loc_type_bottom_field** specify the location of chroma samples as follows:

- If **chroma_format_idc** is equal to 1 (4:2:0 chroma format), **chroma_sample_loc_type_top_field** and **chroma_sample_loc_type_bottom_field** specify the location of chroma samples for the top field and the bottom field, respectively, as shown in Figure E.1.
- Otherwise (**chroma_format_idc** is not equal to 1), the values of the syntax elements **chroma_sample_loc_type_top_field** and **chroma_sample_loc_type_bottom_field** shall be ignored. When **chroma_format_idc** is equal to 2 (4:2:2 chroma format) or 3 (4:4:4 chroma format), the location of chroma samples is specified in clause 6.2. When **chroma_format_idc** is equal to 0, there is no chroma sample array.

The value of **chroma_sample_loc_type_top_field** and **chroma_sample_loc_type_bottom_field** shall be in the range of 0 to 5, inclusive. When the **chroma_sample_loc_type_top_field** and **chroma_sample_loc_type_bottom_field** are not present, the values of **chroma_sample_loc_type_top_field** and **chroma_sample_loc_type_bottom_field** are inferred to be equal to 0.

NOTE 14 – When coding progressive source material, **chroma_sample_loc_type_top_field** and **chroma_sample_loc_type_bottom_field** should have the same value.



Interpretation of symbols

Luma sample position indications:

⊗ Luma sample top field □ Luma sample bottom field

Chroma sample position indications, where gray fill indicates a bottom field sample type and no fill indicates a top field sample type:

○ Chroma sample type 2 ○ Chroma sample type 3
 ○ Chroma sample type 0 ▽ Chroma sample type 1
 ◇ Chroma sample type 4 △ Chroma sample type 5

Figure E.1 – Location of chroma samples for top and bottom fields for **chroma_format_idc equal to 1 (4:2:0 chroma format) as a function of **chroma_sample_loc_type_top_field** and **chroma_sample_loc_type_bottom_field****

Figure E.2 illustrates the indicated relative position of the top-left chroma sample when **chroma_format_idc** is equal to 1 (4:2:0 chroma format), and **chroma_sample_loc_type_top_field** and **chroma_sample_loc_type_bottom_field** are both

equal to the value of a variable ChromaLocType. The region represented by the top-left 4:2:0 chroma sample (depicted as a large red square with a large red dot at its centre) is shown relative to the region represented by the top-left luma sample (depicted as a small black square with a small black dot at its centre). The regions represented by neighbouring luma samples are depicted as small grey squares with small grey dots at their centres.

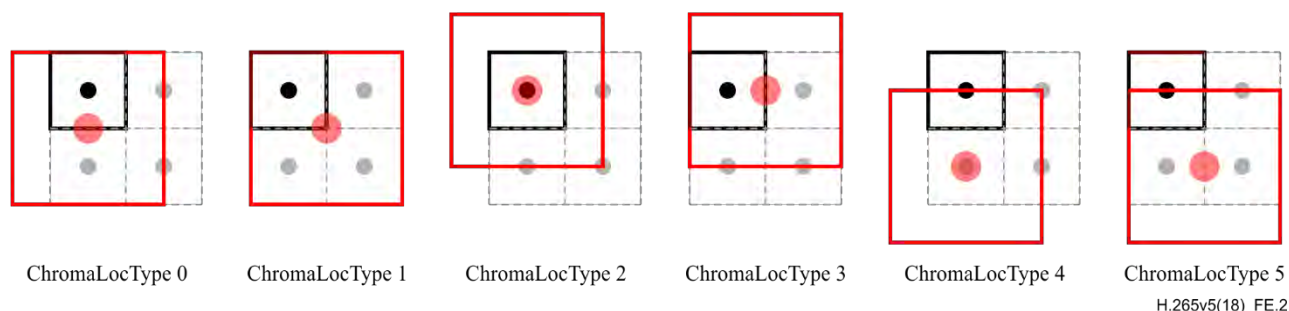


Figure E.2 – Location of the top-left chroma sample when chroma_format_idc is equal to 1 (4:2:0 chroma format) as a function of ChromaLocType

The relative spatial positioning of the chroma samples, as illustrated in Figure E.3, can be expressed by defining two variables HorizontalOffsetC and VerticalOffsetC as a function of chroma_format_idc and the variable ChromaLocType as given by Table E.6, where HorizontalOffsetC is the horizontal (x) position of the centre of the top-left chroma sample relative to the centre of the top-left luma sample in units of luma samples and VerticalOffsetC is the vertical (y) position of the centre of the top-left chroma sample relative to the centre of the top-left luma sample in units of luma samples.

In a typical FIR filter design, when chroma_format_idc is equal to 1 (4:2:0 chroma format) or 2 (4:2:2 chroma format), HorizontalOffsetC and VerticalOffsetC would serve as the phase offsets for the horizontal and vertical filter operations, respectively, for separable downsampling from 4:4:4 chroma format to the chroma format indicated by chroma_format_idc.

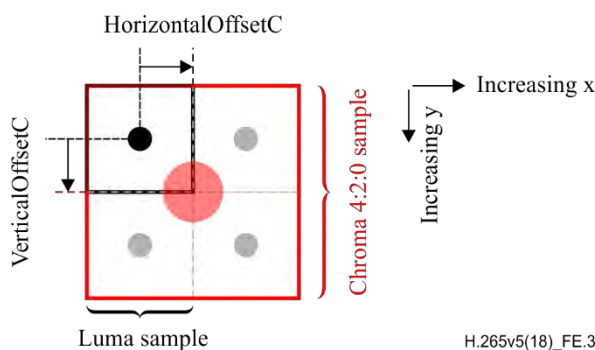


Figure E.3 – Location of the top-left chroma sample when chroma_format_idc is equal to 1 (4:2:0 chroma format) when ChromaLocType is equal to 1

Table E.6 – Definition of HorizontalOffsetC and VerticalOffsetC as a function of chroma_format_idc and ChromaLocType

chroma_format_idc	ChromaLocType	HorizontalOffsetC	VerticalOffsetC
1 (4:2:0)	0	0	0.5
1 (4:2:0)	1	0.5	0.5
1 (4:2:0)	2	0	0
1 (4:2:0)	3	0.5	0
1 (4:2:0)	4	0	1
1 (4:2:0)	5	0.5	1
2 (4:2:2)	–	0	0
3 (4:4:4)	–	0	0

When `chroma_format_idc` is equal to 1 (4:2:0 chroma format) and the decoded video content is intended for interpretation according to Rec. ITU-R BT.2020-2 or Rec. ITU-R BT.2100-2, `chroma_loc_info_present_flag` should be equal to 1, and `chroma_sample_loc_type_top_field` and `chroma_sample_loc_type_bottom_field` should both be equal to 2.

neutral_chroma_indication_flag equal to 1 indicates that the value of all decoded chroma samples is equal to $1 \ll (\text{BitDepth}_C - 1)$. `neutral_chroma_indication_flag` equal to 0 provides no indication of decoded chroma sample values. When `neutral_chroma_indication_flag` is equal to 1, it is a requirement of bitstream conformance that the value of all decoded chroma samples produced by the decoding process shall be equal to $1 \ll (\text{BitDepth}_C - 1)$. When `neutral_chroma_indication_flag` is not present, it is inferred to be equal to 0.

NOTE 15 – When `neutral_chroma_indication_flag` is equal to 1, it is not necessary for the decoder to apply the specified decoding process in order to determine the value of the decoded chroma samples.

field_seq_flag equal to 1 indicates that the CVS conveys pictures that represent fields, and specifies that a picture timing SEI message shall be present in every access unit of the current CVS. `field_seq_flag` equal to 0 indicates that the CVS conveys pictures that represent frames and that a picture timing SEI message may or may not be present in any access unit of the current CVS. When `field_seq_flag` is not present, it is inferred to be equal to 0. When `general_frame_only_constraint_flag` is equal to 1, the value of `field_seq_flag` shall be equal to 0.

NOTE 16 – The specified decoding process does not treat access units conveying pictures that represent fields or frames differently. A sequence of pictures that represent fields would therefore be coded with the picture dimensions of an individual field. For example, access units containing pictures that represent 1080i fields would commonly have cropped output dimensions of 1920x540, while the sequence picture rate would commonly express the rate of the source fields (typically between 50 and 60 Hz), instead of the source frame rate (typically between 25 and 30 Hz).

frame_field_info_present_flag equal to 1 specifies that picture timing SEI messages are present for every picture and include the `pic_struct`, `source_scan_type` and `duplicate_flag` syntax elements. `frame_field_info_present_flag` equal to 0 specifies that the `pic_struct` syntax element is not present in picture timing SEI messages.

When `frame_field_info_present_flag` is present and either or both of the following conditions are true, `frame_field_info_present_flag` shall be equal to 1:

- `field_seq_flag` is equal to 1.
- `general_progressive_source_flag` is equal to 1 and `general_interlaced_source_flag` is equal to 1.

When `frame_field_info_present_flag` is not present, its value is inferred as follows:

- If `general_progressive_source_flag` is equal to 1 and `general_interlaced_source_flag` is equal to 1, `frame_field_info_present_flag` is inferred to be equal to 1.
- Otherwise, `frame_field_info_present_flag` is inferred to be equal to 0.

default_display_window_flag equal to 1 indicates that the default display window parameters follow next in the VUI. `default_display_window_flag` equal to 0 indicates that the default display window parameters are not present. The default display window parameters identify the area that is within the conformance cropping window and that is suggested to be displayed in the absence of any alternative indication (provided within the bitstream or by external means not specified in this Specification) of preferred display characteristics.

def_disp_win_left_offset, **def_disp_win_right_offset**, **def_disp_win_top_offset** and **def_disp_win_bottom_offset** specify the samples of the pictures in the CVS that are within the default display window, in terms of a rectangular region specified in picture coordinates for display. When `default_display_window_flag` is equal to 0, the values of `def_disp_win_left_offset`, `def_disp_win_right_offset`, `def_disp_win_top_offset` and `def_disp_win_bottom_offset` are inferred to be equal to 0.

The following variables are derived from the default display window parameters:

$$\text{leftOffset} = \text{conf_win_left_offset} + \text{def_disp_win_left_offset} \quad (\text{E-68})$$

$$\text{rightOffset} = \text{conf_win_right_offset} + \text{def_disp_win_right_offset} \quad (\text{E-69})$$

$$\text{topOffset} = \text{conf_win_top_offset} + \text{def_disp_win_top_offset} \quad (\text{E-70})$$

$$\text{bottomOffset} = \text{conf_win_bottom_offset} + \text{def_disp_win_bottom_offset} \quad (\text{E-71})$$

The default display window contains the luma samples with horizontal picture coordinates from $\text{SubWidthC} * \text{leftOffset}$ to $\text{pic_width_in_luma_samples} - (\text{SubWidthC} * \text{rightOffset} + 1)$ and vertical picture coordinates from $\text{SubHeightC} * \text{topOffset}$ to $\text{pic_height_in_luma_samples} - (\text{SubHeightC} * \text{bottomOffset} + 1)$, inclusive.

The value of $\text{SubWidthC} * (\text{leftOffset} + \text{rightOffset})$ shall be less than $\text{pic_width_in_luma_samples}$ and the value of $\text{SubHeightC} * (\text{topOffset} + \text{bottomOffset})$ shall be less than $\text{pic_height_in_luma_samples}$.

When ChromaArrayType is not equal to 0, the corresponding specified samples of the two chroma arrays are the samples having picture coordinates $(x / \text{SubWidthC}, y / \text{SubHeightC})$, where (x, y) are the picture coordinates of the specified luma samples.

vui_timing_info_present_flag equal to 1 specifies that **vui_num_units_in_tick**, **vui_time_scale**, **vui_poc_proportional_to_timing_flag** and **vui_hrd_parameters_present_flag** are present in the **vui_parameters()** syntax structure. **vui_timing_info_present_flag** equal to 0 specifies that **vui_num_units_in_tick**, **vui_time_scale**, **vui_poc_proportional_to_timing_flag** and **vui_hrd_parameters_present_flag** are not present in the **vui_parameters()** syntax structure.

vui_num_units_in_tick is the number of time units of a clock operating at the frequency **vui_time_scale** Hz that corresponds to one increment (called a clock tick) of a clock tick counter. **vui_num_units_in_tick** shall be greater than 0. A clock tick, in units of seconds, is equal to the quotient of **vui_num_units_in_tick** divided by **vui_time_scale**. For example, when the picture rate of a video signal is 25 Hz, **vui_time_scale** may be equal to 27 000 000 and **vui_num_units_in_tick** may be equal to 1 080 000 and consequently a clock tick may be equal to 0.04 seconds.

When **vps_num_units_in_tick** is present in the VPS referred to by the SPS, **vui_num_units_in_tick**, when present, shall be equal to **vps_num_units_in_tick**, and when not present, is inferred to be equal to **vps_num_units_in_tick**.

vui_time_scale is the number of time units that pass in one second. For example, a time coordinate system that measures time using a 27 MHz clock has a **vui_time_scale** of 27 000 000. The value of **vui_time_scale** shall be greater than 0.

When **vps_time_scale** is present in the VPS referred to by the SPS, **vui_time_scale**, when present, shall be equal to **vps_time_scale**, and when not present, is inferred to be equal to **vps_time_scale**.

vui_poc_proportional_to_timing_flag equal to 1 indicates that the picture order count value for each picture in the CVS that is not the first picture in the CVS, in decoding order, is proportional to the output time of the picture relative to the output time of the first picture in the CVS. **vui_poc_proportional_to_timing_flag** equal to 0 indicates that the picture order count value for each picture in the CVS that is not the first picture in the CVS, in decoding order, may or may not be proportional to the output time of the picture relative to the output time of the first picture in the CVS.

When **vps_poc_proportional_to_timing_flag** is present in the VPS referred to by the SPS and the value is equal to 1, **vui_poc_proportional_to_timing_flag**, when present, shall be equal to 1.

vui_num_ticks_poc_diff_one_minus1 plus 1 specifies the number of clock ticks corresponding to a difference of picture order count values equal to 1. The value of **vui_num_ticks_poc_diff_one_minus1** shall be in the range of 0 to $2^{32} - 2$, inclusive.

When **vps_num_ticks_poc_diff_one_minus1** is present in the VPS referred to by the SPS, **vui_num_ticks_poc_diff_one_minus1**, when present, shall be equal to **vps_num_ticks_poc_diff_one_minus1**.

vui_hrd_parameters_present_flag equal to 1 specifies that the syntax structure **hrd_parameters()** is present in the **vui_parameters()** syntax structure. **vui_hrd_parameters_present_flag** equal to 0 specifies that the syntax structure **hrd_parameters()** is not present in the **vui_parameters()** syntax structure.

bitstream_restriction_flag equal to 1, specifies that the bitstream restriction parameters for the CVS are present. **bitstream_restriction_flag** equal to 0, specifies that the bitstream restriction parameters for the CVS are not present.

tiles_fixed_structure_flag equal to 1 indicates that each PPS that is active in the CVS has the same value of the syntax elements **num_tile_columns_minus1**, **num_tile_rows_minus1**, **uniform_spacing_flag**, **column_width_minus1[i]**, **row_height_minus1[i]** and **loop_filter_across_tiles_enabled_flag**, when present. **tiles_fixed_structure_flag** equal to 0 indicates that tiles syntax elements in different PPSs may or may not have the same value. When the **tiles_fixed_structure_flag** syntax element is not present, it is inferred to be equal to 0.

NOTE 17 – The signalling of **tiles_fixed_structure_flag** equal to 1 is a guarantee to a decoder that each picture in the CVS has the same number of tiles distributed in the same way which might be useful for workload allocation in the case of multi-threaded decoding.

motion_vectors_over_pic_boundaries_flag equal to 0 indicates that no sample outside the picture boundaries and no sample at a fractional sample position for which the sample value is derived using one or more samples outside the picture boundaries is used for inter prediction of any sample. **motion_vectors_over_pic_boundaries_flag** equal to 1 indicates that one or more samples outside the picture boundaries may be used in inter prediction. When the **motion_vectors_over_pic_boundaries_flag** syntax element is not present, **motion_vectors_over_pic_boundaries_flag** value is inferred to be equal to 1.

restricted_ref_pic_lists_flag equal to 1 indicates that all P and B slices (when present) that belong to the same picture have an identical reference picture list 0 and that all B slices (when present) that belong to the same picture have an identical reference picture list 1.

min_spatial_segmentation_idc, when not equal to 0, establishes a bound on the maximum possible size of distinct coded spatial segmentation regions in the pictures of the CVS. When **min_spatial_segmentation_idc** is not present, it is inferred to be equal to 0. The value of **min_spatial_segmentation_idc** shall be in the range of 0 to 4095, inclusive.

The variable **minSpatialSegmentationTimes4** is derived from **min_spatial_segmentation_idc** as follows:

$$\text{minSpatialSegmentationTimes4} = \text{min_spatial_segmentation_idc} + 4 \quad (\text{E-72})$$

A slice is said to contain a specific luma sample when the coding block that contains the luma sample is contained in the slice. Correspondingly, a tile is said to contain a specific luma sample when the coding block that contains the luma sample is contained in the tile.

Depending on the value of **min_spatial_segmentation_idc**, the following applies:

- If **min_spatial_segmentation_idc** is equal to 0, no limit on the maximum size of spatial segments is indicated.
- Otherwise (**min_spatial_segmentation_idc** is not equal to 0), it is a requirement of bitstream conformance that exactly one of the following conditions shall be true:
 - In each PPS that is activated within the CVS, **tiles_enabled_flag** is equal to 0 and **entropy_coding_sync_enabled_flag** is equal to 0 and there is no slice in the CVS that contains more than $(4 * \text{PicSizeInSamplesY}) / \text{minSpatialSegmentationTimes4}$ luma samples.
 - In each PPS that is activated within the CVS, **tiles_enabled_flag** is equal to 1 and **entropy_coding_sync_enabled_flag** is equal to 0 and there is no tile in the CVS that contains more than $(4 * \text{PicSizeInSamplesY}) / \text{minSpatialSegmentationTimes4}$ luma samples.
 - In each PPS that is activated within the CVS, **tiles_enabled_flag** is equal to 0 and **entropy_coding_sync_enabled_flag** is equal to 1 and the syntax elements **pic_width_in_luma_samples**, **pic_height_in_luma_samples** and the variable **CtbSizeY** obey the following constraint:

$$(2 * \text{pic_height_in_luma_samples} + \text{pic_width_in_luma_samples}) * \text{CtbSizeY} \leq (4 * \text{PicSizeInSamplesY}) / \text{minSpatialSegmentationTimes4} \quad (\text{E-73})$$

NOTE 18 – The syntax element **min_spatial_segmentation_idc** can be used by a decoder to calculate the maximum number of luma samples to be processed by one processing thread, making the assumption that the decoder maximally utilizes the parallel decoding information. However, it is important to be aware that there may be some inter-dependencies between the different threads – e.g., due to entropy coding synchronization or deblocking filtering across tile or slice boundaries. To aid decoders in planning the decoding workload distribution, encoders are encouraged to set the value of **min_spatial_segmentation_idc** to the highest possible value for which one of the above three conditions is true. For example, for the case when **tiles_enabled_flag** is equal to 0 and **entropy_coding_sync_enabled_flag** is equal to 1, encoders should set **min_spatial_segmentation_idc** equal to $4 * \text{PicSizeInSamplesY} / ((2 * \text{pic_height_in_luma_samples} + \text{pic_width_in_luma_samples}) * \text{CtbSizeY}) - 4$.

max_bytes_per_pic_denom indicates a number of bytes not exceeded by the sum of the sizes of the VCL NAL units associated with any coded picture in the CVS.

The number of bytes that represent a picture in the NAL unit stream is specified for this purpose as the total number of bytes of VCL NAL unit data (i.e., the total of the **NumBytesInNalUnit** variables for the VCL NAL units) for the picture. The value of **max_bytes_per_pic_denom** shall be in the range of 0 to 16, inclusive.

Depending on the value of **max_bytes_per_pic_denom** the following applies:

- If **max_bytes_per_pic_denom** is equal to 0, no limits are indicated.
- Otherwise (**max_bytes_per_pic_denom** is not equal to 0), it is a requirement of bitstream conformance that no coded picture shall be represented in the CVS by more than the following number of bytes.

$$(\text{PicSizeInMinCbsY} * \text{RawMinCuBits}) \div (8 * \text{max_bytes_per_pic_denom}) \quad (\text{E-74})$$

When the **max_bytes_per_pic_denom** syntax element is not present, the value of **max_bytes_per_pic_denom** is inferred to be equal to 2.

max_bits_per_min_cu_denom indicates an upper bound for the number of coded bits of **coding_unit()** data for any coding block in any picture of the CVS. The value of **max_bits_per_min_cu_denom** shall be in the range of 0 to 16, inclusive.

Depending on the value of `max_bits_per_min_cu_denom`, the following applies:

- If `max_bits_per_min_cu_denom` is equal to 0, no limit is specified by this syntax element.
- Otherwise (`max_bits_per_min_cu_denom` is not equal to 0), it is a requirement of bitstream conformance that no coded coding_unit() shall be represented in the bitstream by more than the following number of bits:

$$(128 + \text{RawMinCuBits}) \div \text{max_bits_per_min_cu_denom} \\ * (1 \ll (2 * (\log_2 \text{CbSize} - \text{MinCbLog}_2 \text{SizeY}))) \quad (\text{E-75})$$

where `log2CbSize` is the value of `log2CbSize` for the given coding block and the number of bits of coding_unit() data for the same coding block is given by the number of times `read_bits(1)` is called in clauses 9.3.4.3.3 and 9.3.4.3.4.

When the `max_bits_per_min_cu_denom` is not present, the value of `max_bits_per_min_cu_denom` is inferred to be equal to 1.

log2_max_mv_length_horizontal and **log2_max_mv_length_vertical** indicate the maximum absolute value of a decoded horizontal and vertical motion vector component, respectively, in quarter luma sample units, for all pictures in the CVS. A value of `n` asserts that no value of a motion vector component is outside the range of -2^n to $2^n - 1$, inclusive, in units of quarter luma sample displacement, where `n` refers to the value of `log2_max_mv_length_horizontal` and `log2_max_mv_length_vertical` for the horizontal and vertical component of the MV, respectively. The values of `log2_max_mv_length_horizontal` and `log2_max_mv_length_vertical` shall be in the range of 0 to 15, inclusive. When not present, the values of `log2_max_mv_length_horizontal` and `log2_max_mv_length_vertical` are both inferred to be equal to 15.

E.3.2 HRD parameters semantics

The `hrd_parameters()` syntax structure provides HRD parameters used in the HRD operations for a layer set. When the `hrd_parameters()` syntax structure is included in a VPS, the applicable layer set to which the `hrd_parameters()` syntax structure applies is specified by the corresponding `hrd_layer_set_idx[i]` syntax element in the VPS. When the `hrd_parameters()` syntax structure is included in an SPS, the layer set to which the `hrd_parameters()` syntax structure applies is the layer set for which the associated layer identifier list contains all `nuh_layer_id` values present in the CVS.

For interpretation of the following semantics, the bitstream (or a part thereof) refers to the bitstream subset (or a part thereof) associated with the layer set to which the `hrd_parameters()` syntax structure applies.

nal_hrd_parameters_present_flag equal to 1 specifies that NAL HRD parameters (pertaining to the Type II bitstream conformance point) are present in the `hrd_parameters()` syntax structure. **nal_hrd_parameters_present_flag** equal to 0 specifies that NAL HRD parameters are not present in the `hrd_parameters()` syntax structure.

NOTE 1 – When `nal_hrd_parameters_present_flag` is equal to 0, the conformance of the bitstream cannot be verified without provision of the NAL HRD parameters and all buffering period SEI messages, and, when `vcl_hrd_parameters_present_flag` is also equal to 0, all picture timing and decoding unit information SEI messages, by some means not specified in this Specification.

The variable `NalHrdBpPresentFlag` is derived as follows:

- If one or more of the following conditions are true, the value of `NalHrdBpPresentFlag` is set equal to 1:
 - `nal_hrd_parameters_present_flag` is present in the bitstream and is equal to 1.
 - The need for the presence of buffering period parameters for NAL HRD operation in the bitstream in buffering period SEI messages is determined by the application, by some means not specified in this Specification.
- Otherwise, the value of `NalHrdBpPresentFlag` is set equal to 0.

vcl_hrd_parameters_present_flag equal to 1 specifies that VCL HRD parameters (pertaining to the Type I bitstream conformance point) are present in the `hrd_parameters()` syntax structure. **vcl_hrd_parameters_present_flag** equal to 0 specifies that VCL HRD parameters are not present in the `hrd_parameters()` syntax structure.

NOTE 2 – When `vcl_hrd_parameters_present_flag` is equal to 0, the conformance of the bitstream cannot be verified without provision of the VCL HRD parameters and all buffering period SEI messages, and, when `nal_hrd_parameters_present_flag` is also equal to 0, all picture timing and decoding unit information SEI messages, by some means not specified in this Specification.

The variable `VclHrdBpPresentFlag` is derived as follows:

- If one or more of the following conditions are true, the value of `VclHrdBpPresentFlag` is set equal to 1:
 - `vcl_hrd_parameters_present_flag` is present in the bitstream and is equal to 1.
 - The need for the presence of buffering period parameters for VCL HRD operation in the bitstream in buffering period SEI messages is determined by the application, by some means not specified in this Specification.
- Otherwise, the value of `VclHrdBpPresentFlag` is set equal to 0.

The variable CpbDpbDelaysPresentFlag is derived as follows:

- If one or more of the following conditions are true, the value of CpbDpbDelaysPresentFlag is set equal to 1:
 - nal_hrd_parameters_present_flag is present in the bitstream and is equal to 1.
 - vcl_hrd_parameters_present_flag is present in the bitstream and is equal to 1.
 - The need for presence of CPB and DPB output delays to be present in the bitstream in picture timing SEI messages is determined by the application, by some means not specified in this Specification.
- Otherwise, the value of CpbDpbDelaysPresentFlag is set equal to 0.

When NalHrdBpPresentFlag and VclHrdBpPresentFlag are both equal to 0, the value of CpbDpbDelaysPresentFlag shall be equal to 0.

sub_pic_hrd_params_present_flag equal to 1 specifies that sub-picture level HRD parameters are present and the HRD may operate at access unit level or sub-picture level. **sub_pic_hrd_params_present_flag** equal to 0 specifies that sub-picture level HRD parameters are not present and the HRD operates at access unit level. When **sub_pic_hrd_params_present_flag** is not present, its value is inferred to be equal to 0.

tick_divisor_minus2 is used to specify the clock sub-tick. A clock sub-tick is the minimum interval of time that can be represented in the coded data when **sub_pic_hrd_params_present_flag** is equal to 1.

du_cpb_removal_delay_increment_length_minus1 plus 1 specifies the length, in bits, of the **du_cpb_removal_delay_increment_minus1[i]** and **du_common_cpb_removal_delay_increment_minus1** syntax elements of the picture timing SEI message and the **du_spt_cpb_removal_delay_increment** syntax element in the decoding unit information SEI message.

sub_pic_cpb_params_in_pic_timing_sei_flag equal to 1 specifies that sub-picture level CPB removal delay parameters are present in picture timing SEI messages and no decoding unit information SEI message is available (in the CVS or provided through external means not specified in this Specification). **sub_pic_cpb_params_in_pic_timing_sei_flag** equal to 0 specifies that sub-picture level CPB removal delay parameters are present in decoding unit information SEI messages and picture timing SEI messages do not include sub-picture level CPB removal delay parameters. When the **sub_pic_cpb_params_in_pic_timing_sei_flag** syntax element is not present, it is inferred to be equal to 0.

dpb_output_delay_du_length_minus1 plus 1 specifies the length, in bits, of the **pic_dpb_output_du_delay** syntax element in the picture timing SEI message and the **pic_spt_dpb_output_du_delay** syntax element in the decoding unit information SEI message.

bit_rate_scale (together with **bit_rate_value_minus1[i]**) specifies the maximum input bit rate of the i-th CPB.

cpb_size_scale (together with **cpb_size_value_minus1[i]**) specifies the CPB size of the i-th CPB when the CPB operates at the access unit level.

cpb_size_du_scale (together with **cpb_size_du_value_minus1[i]**) specifies the CPB size of the i-th CPB when the CPB operates at sub-picture level.

initial_cpb_removal_delay_length_minus1 plus 1 specifies the length, in bits, of the **nal_initial_cpb_removal_delay[i]**, **nal_initial_cpb_removal_offset[i]**, **vcl_initial_cpb_removal_delay[i]** and **vcl_initial_cpb_removal_offset[i]** syntax elements of the buffering period SEI message. When the **initial_cpb_removal_delay_length_minus1** syntax element is not present, it is inferred to be equal to 23.

au_cpb_removal_delay_length_minus1 plus 1 specifies the length, in bits, of the **cpb_delay_offset** syntax element in the buffering period SEI message and the **au_cpb_removal_delay_minus1** syntax element in the picture timing SEI message. When the **au_cpb_removal_delay_length_minus1** syntax element is not present, it is inferred to be equal to 23.

dpb_output_delay_length_minus1 plus 1 specifies the length, in bits, of the **dpb_delay_offset** syntax element in the buffering period SEI message and the **pic_dpb_output_delay** syntax element in the picture timing SEI message. When the **dpb_output_delay_length_minus1** syntax element is not present, it is inferred to be equal to 23.

fixed_pic_rate_general_flag[i] equal to 1 indicates that, when HighestTid is equal to i, the temporal distance between the HRD output times of consecutive pictures in output order is constrained as specified below. **fixed_pic_rate_general_flag[i]** equal to 0 indicates that this constraint may not apply.

When **fixed_pic_rate_general_flag[i]** is not present, it is inferred to be equal to 0.

fixed_pic_rate_within_cvs_flag[i] equal to 1 indicates that, when HighestTid is equal to i, the temporal distance between the HRD output times of consecutive pictures in output order is constrained as specified below. **fixed_pic_rate_within_cvs_flag[i]** equal to 0 indicates that this constraint may not apply.

When `fixed_pic_rate_general_flag[i]` is equal to 1, the value of `fixed_pic_rate_within_cvs_flag[i]` is inferred to be equal to 1.

elemental_duration_in_tc_minus1[i] plus 1 (when present) specifies, when `HighestTid` is equal to `i`, the temporal distance, in clock ticks, between the elemental units that specify the HRD output times of consecutive pictures in output order as specified below. The value of `elemental_duration_in_tc_minus1[i]` shall be in the range of 0 to 2 047, inclusive.

For each picture `n` that is output and not the last picture in the bitstream (in output order) that is output, the value of the variable `DpbOutputElementalInterval[n]` is specified by:

$$\text{DpbOutputElementalInterval}[n] = \text{DpbOutputInterval}[n] \div \text{DeltaToDivisor} \quad (\text{E-76})$$

where `DpbOutputInterval[n]` is specified in Equation C-16 and `DeltaToDivisor` is specified in Table E.7 based on the value of `frame_field_info_present_flag` and `pic_struct` for the CVS containing picture `n`. Entries marked "-" in Table E.7 indicate a lack of dependence of `DeltaToDivisor` on the corresponding syntax element.

When `HighestTid` is equal to `i` and `fixed_pic_rate_general_flag[i]` is equal to 1 for a CVS containing picture `n`, the value computed for `DpbOutputElementalInterval[n]` shall be equal to `ClockTick * (elemental_duration_in_tc_minus1[i] + 1)`, wherein `ClockTick` is as specified in Equation C-1 (using the value of `ClockTick` for the CVS containing picture `n`) when one of the following conditions is true for the following picture in output order `nextPicInOutputOrder` that is specified for use in Equation C-16:

- picture `nextPicInOutputOrder` is in the same CVS as picture `n`.
- picture `nextPicInOutputOrder` is in a different CVS and `fixed_pic_rate_general_flag[i]` is equal to 1 in the CVS containing picture `nextPicInOutputOrder`, the value of `ClockTick` is the same for both CVSs and the value of `elemental_duration_in_tc_minus1[i]` is the same for both CVSs.

When `HighestTid` is equal to `i` and `fixed_pic_rate_within_cvs_flag[i]` is equal to 1 for a CVS containing picture `n`, the value computed for `DpbOutputElementalInterval[n]` shall be equal to `ClockTick * (elemental_duration_in_tc_minus1[i] + 1)`, wherein `ClockTick` is as specified in Equation C-1 (using the value of `ClockTick` for the CVS containing picture `n`) when the following picture in output order `nextPicInOutputOrder` that is specified for use in Equation C-16 is in the same CVS as picture `n`.

Table E.7 – Divisor for computation of `DpbOutputElementalInterval[n]`

frame_field_info_present_flag	pic_struct	DeltaToDivisor
0	-	1
1	1	1
1	2	1
1	0	1
1	3	2
1	4	2
1	5	3
1	6	3
1	7	2
1	8	3
1	9	1
1	10	1
1	11	1
1	12	1

low_delay_hrd_flag[i] specifies the HRD operational mode, when `HighestTid` is equal to `i`, as specified in Annex C or clause F.13. When not present, the value of `low_delay_hrd_flag[i]` is inferred to be equal to 0.

NOTE 3 – When `low_delay_hrd_flag[i]` is equal to 1, "big pictures" that violate the nominal CPB removal times due to the number of bits used by an access unit are permitted. It is expected, but not required, that such "big pictures" occur only occasionally.

cpb_cnt_minus1[i] plus 1 specifies the number of alternative CPB specifications in the bitstream of the CVS when `HighestTid` is equal to `i`. The value of `cpb_cnt_minus1[i]` shall be in the range of 0 to 31, inclusive. When not present, the value of `cpb_cnt_minus1[i]` is inferred to be equal to 0.

E.3.3 Sub-layer HRD parameters semantics

The variable CpbCnt is set equal to `cpb_cnt_minus1[subLayerId] + 1`.

bit_rate_value_minus1[i] (together with `bit_rate_scale`) specifies the maximum input bit rate for the *i*-th CPB when the CPB operates at the access unit level. `bit_rate_value_minus1[i]` shall be in the range of 0 to $2^{32} - 2$, inclusive. For any *i* > 0, `bit_rate_value_minus1[i]` shall be greater than `bit_rate_value_minus1[i - 1]`.

When `SubPicHrdFlag` is equal to 0, the bit rate in bits per second is given by:

$$\text{BitRate}[i] = (\text{bit_rate_value_minus1}[i] + 1) * 2^{(6 + \text{bit_rate_scale})} \quad (\text{E-77})$$

When `SubPicHrdFlag` is equal to 0 and the `bit_rate_value_minus1[i]` syntax element is not present, the value of `BitRate[i]` is inferred to be equal to `BrVclFactor * MaxBR` for VCL HRD parameters and to be equal to `BrNalFactor * MaxBR` for NAL HRD parameters, where `MaxBR`, `BrVclFactor` and `BrNalFactor` are specified in clause A.4.

cpb_size_value_minus1[i] is used together with `cpb_size_scale` to specify the *i*-th CPB size when the CPB operates at the access unit level. `cpb_size_value_minus1[i]` shall be in the range of 0 to $2^{32} - 2$, inclusive. For any *i* greater than 0, `cpb_size_value_minus1[i]` shall be less than or equal to `cpb_size_value_minus1[i - 1]`.

When `SubPicHrdFlag` is equal to 0, the CPB size in bits is given by:

$$\text{CpbSize}[i] = (\text{cpb_size_value_minus1}[i] + 1) * 2^{(4 + \text{cpb_size_scale})} \quad (\text{E-78})$$

When `SubPicHrdFlag` is equal to 0 and the `cpb_size_value_minus1[i]` syntax element is not present, the value of `CpbSize[i]` is inferred to be equal to `CpbVclFactor * MaxCPB` for VCL HRD parameters and to be equal to `CpbNalFactor * MaxCPB` for NAL HRD parameters, where `MaxCPB`, `CpbVclFactor` and `CpbNalFactor` are specified in clause A.4.

cpb_size_du_value_minus1[i] is used together with `cpb_size_du_scale` to specify the *i*-th CPB size when the CPB operates at sub-picture level. `cpb_size_du_value_minus1[i]` shall be in the range of 0 to $2^{32} - 2$, inclusive. For any *i* greater than 0, `cpb_size_du_value_minus1[i]` shall be less than or equal to `cpb_size_du_value_minus1[i - 1]`.

When `SubPicHrdFlag` is equal to 1, the CPB size in bits is given by:

$$\text{CpbSize}[i] = (\text{cpb_size_du_value_minus1}[i] + 1) * 2^{(4 + \text{cpb_size_du_scale})} \quad (\text{E-79})$$

When `SubPicHrdFlag` is equal to 1 and the `cpb_size_du_value_minus1[i]` syntax element is not present, the value of `CpbSize[i]` is inferred to be equal to `CpbVclFactor * MaxCPB` for VCL HRD parameters and to be equal to `CpbNalFactor * MaxCPB` for NAL HRD parameters, where `MaxCPB`, `CpbVclFactor` and `CpbNalFactor` are specified in clause A.4.

bit_rate_du_value_minus1[i] (together with `bit_rate_scale`) specifies the maximum input bit rate for the *i*-th CPB when the CPB operates at the sub-picture level. `bit_rate_du_value_minus1[i]` shall be in the range of 0 to $2^{32} - 2$, inclusive. For any *i* > 0, `bit_rate_du_value_minus1[i]` shall be greater than `bit_rate_du_value_minus1[i - 1]`.

When `SubPicHrdFlag` is equal to 1, the bit rate in bits per second is given by:

$$\text{BitRate}[i] = (\text{bit_rate_du_value_minus1}[i] + 1) * 2^{(6 + \text{bit_rate_scale})} \quad (\text{E-80})$$

When `SubPicHrdFlag` is equal to 1 and the `bit_rate_du_value_minus1[i]` syntax element is not present, the value of `BitRate[i]` is inferred to be equal to `BrVclFactor * MaxBR` for VCL HRD parameters and to be equal to `BrNalFactor * MaxBR` for NAL HRD parameters, where `MaxBR`, `BrVclFactor` and `BrNalFactor` are specified in clause A.4.

cbr_flag[i] equal to 0 specifies that to decode this CVS by the HRD using the *i*-th CPB specification, the hypothetical stream scheduler (HSS) operates in an intermittent bit rate mode. `cbr_flag[i]` equal to 1 specifies that the HSS operates in a constant bit rate (CBR) mode. When not present, the value of `cbr_flag[i]` is inferred to be equal to 0.

Annex F

Common specifications for multi-layer extensions

(This annex forms an integral part of this Recommendation | International Standard.)

F.1 Scope

This annex specifies the common syntax, semantics and decoding processes for multi-layer video coding extensions, with references made to clauses 2-9 and Annexes A-E, G and H.

F.2 Normative references

The list of normative references in clause 2 applies.

F.3 Definitions

For the purpose of this annex, the following definitions apply in addition to the definitions in clause 3. These definitions are either not present in clause 3 or replace definitions in clause 3.

- F.3.1 access unit:** A set of *NAL units* that are associated with each other according to a specified classification rule, are consecutive in *decoding order* and contain at most one *coded picture* with any specific value of *nuh_layer_id*.
- F.3.2 additional layer set:** A set of *layers* of a *bitstream* with a set of *layers* of one or more *non-base layer subtrees*.
- F.3.3 alternative output layer:** A *layer* that is a *reference layer* of an *output layer* and which may include a *picture* that may be output when no *picture* of the *output layer* is present in the *access unit* containing the *picture*.
- F.3.4 associated IRAP picture:** The previous *IRAP picture* in *decoding order* within the same *layer* (if present).
- F.3.5 auxiliary picture:** A *picture* that has no normative effect on the *decoding process* of *primary pictures* and with a *nuh_layer_id* value such that *AuxId[nuh_layer_id]* is greater than 0.
- F.3.6 auxiliary picture layer:** A *layer* with a *nuh_layer_id* value such that *AuxId[nuh_layer_id]* is greater than 0.
- F.3.7 base bitstream partition:** A *bitstream partition* that contains the *layer* with *nuh_layer_id* equal to *SmallestLayerId*.
- F.3.8 bitstream partition:** A sequence of bits, in the form of a *NAL unit stream* or a *byte stream*, that is a subset of a *bitstream* according to a *partitioning scheme*.
- F.3.9 broken link access (BLA) access unit:** An *access unit* in which the *coded picture* with *nuh_layer_id* equal to *SmallestLayerId* is a *BLA picture*.
- F.3.10 clean random access (CRA) access unit:** An *access unit* in which the *coded picture* with *nuh_layer_id* equal to *SmallestLayerId* is a *CRA picture*.
- F.3.11 coded layer-wise video sequence (CLVS):** A sequence of *coded pictures*, with the same *nuh_layer_id* value *nuhLayerId*, that consists, in decoding order, of an *IRAP picture* with *NoRaslOutputFlag* equal to 1 or a *picture* with *FirstPicInLayerDecodedFlag[nuhLayerId]* equal to 0, followed by all *coded pictures*, if any, up to but excluding the next *IRAP picture* with *NoRaslOutputFlag* equal to 1 or the next *picture* with *FirstPicInLayerDecodedFlag[nuhLayerId]* equal to 0.
- F.3.12 coded picture:** A *coded representation* of a *picture* comprising *VCL NAL units* with a particular value of *nuh_layer_id* within an *access unit* and containing all *CTUs* of the *picture*.
- F.3.13 coded picture buffer (CPB):** A first-in first-out buffer containing *decoding units* in *decoding order* specified in the *hypothetical reference decoder* in Annex C or in clause F.13.
- F.3.14 coded video sequence (CVS):** A sequence of *access units* that consists, in decoding order, of an *initial IRAP access unit*, followed by zero or more *access units* that are not *initial IRAP access units*, including all subsequent *access units* up to but not including any subsequent *access unit* that is an *initial IRAP access unit*.
- F.3.15 collocated sample:** A reference picture sample located at a corresponding position to the current sample as determined through the resampling process, for use in *inter-layer prediction*.
- F.3.16 cross-layer random access skip (CL-RAS) picture:** A *picture* with *nuh_layer_id* equal to *layerId* such that *LayerInitializedFlag[layerId]* is equal to 0 when the decoding process for starting the decoding of a coded picture with *nuh_layer_id* greater than 0 is invoked.

- F.3.17 decoded picture buffer (DPB):** A buffer holding *decoded pictures* for reference, output reordering, or output delay specified for the *hypothetical reference decoder* in Annex C or in clause F.13.
- F.3.18 decoding unit:** A *partition unit* if SubPicHrdFlag is equal to 0 or a subset of a *partition unit* otherwise, consisting of one or more *VCL NAL units* in a *partition unit* and the *associated non-VCL NAL units*.
- F.3.19 direct predicted layer:** A *layer* for which another *layer* is a *direct reference layer*.
- F.3.20 direct reference layer:** A *layer* that may be used for *inter-layer prediction* of another *layer*.
- F.3.21 direct reference layer picture:** A *picture* in a *direct reference layer* that is used as input to derive an *inter-layer reference picture* for *inter-layer prediction* of the current *picture* and is in the same *access unit* as the current *picture*.
- F.3.22 independent layer:** A *layer* that does not have *direct reference layers*.
- F.3.23 indirect predicted layer:** A *layer* for which another *layer* is an *indirect reference layer*.
- F.3.24 indirect reference layer:** A *layer* that is not a *direct reference layer* of a second *layer* but is a *direct reference layer* of a third *layer* that is a *reference layer* of the second *layer*.
- F.3.25 initial intra random access point (IRAP) access unit:** An *IRAP access unit* in which the *coded picture* with nuh_layer_id equal to SmallestLayerId has NoRaslOutputFlag equal to 1.
- F.3.26 inter-layer prediction:** A *prediction* in a manner that is dependent on data elements (e.g., sample values or motion vectors) of *reference pictures* with a different value of nuh_layer_id than that of the current *picture*.
- F.3.27 inter-layer reference picture:** A *reference picture* that may be used for *inter-layer prediction* of the current *picture* and is in the same *access unit* as the current *picture*.
- F.3.28 intra random access point (IRAP) access unit:** An *access unit* in which the *coded picture* with nuh_layer_id equal to SmallestLayerId is an *IRAP picture*.
- F.3.29 intra random access point (IRAP) picture:** A *coded picture* for which each *VCL NAL unit* has nal_unit_type in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive.
- NOTE – An IRAP picture belonging to an independent layer contains only I slices. An IRAP picture belonging to a predicted layer with nuh_layer_id value currLayerId may contain P, B and I slices, cannot use inter prediction from other pictures with nuh_layer_id equal to currLayerId and may use inter-layer prediction from its direct reference layers. The IRAP picture belonging to a predicted layer with nuh_layer_id value currLayerId and all subsequent non-RASL pictures with nuh_layer_id equal to currLayerId in decoding order can be correctly decoded without performing the decoding process of any pictures with nuh_layer_id equal to currLayerId that precede the IRAP picture in decoding order, when the necessary parameter sets are available when they need to be activated and LayerInitializedFlag[refLayerId] is equal to 1 for refLayerId equal to all nuh_layer_id values of the direct reference layers of the layer with nuh_layer_id equal to currLayerId.
- F.3.30 layer subtree:** A subset of the *layers* of a *layer tree* including all the *reference layers* of the *layers* within the subset.
- F.3.31 layer tree:** A set of *layers* such that each *layer* in the set of *layers* is a *predicted layer* or a *reference layer* of at least one other *layer* in the set of *layers* and no *layer* outside the set of *layers* is a *predicted layer* or a *reference layer* of any *layer* in the set of *layers*.
- F.3.32 leading picture:** A *picture* that is in the same *layer* as the *associated IRAP picture* and precedes the *associated IRAP picture* in *output order*.
- F.3.33 necessary layer:** A *layer* in an *output operation point* associated with an *OLS*, the *layer* being an *output layer* of the *OLS*, or a *reference layer* of an *output layer* of the *OLS*.
- F.3.34 non-base layer:** A *layer* in which all *VCL NAL units* have the same nuh_layer_id value greater than 0.
- F.3.35 non-base layer subtree:** A *layer subtree* that does not include the *base layer*.
- F.3.36 output layer:** A *layer* of an *output layer set* that is output when TargetOlsIdx is equal to the index of the *output layer set*.
- F.3.37 output layer set (OLS):** A set of *layers* consisting of the *layers* of one of the specified *layer sets*, where one or more *layers* in the set of *layers* are indicated to be *output layers*.
- F.3.38 output operation point:** A *bitstream* that is created from an input *bitstream* by operation of the *sub-bitstream extraction process* with the input *bitstream*, a target highest TemporalId and a target layer identifier list as inputs, and that is associated with a set of *output layers*.

- F.3.39 output order:** The order in which the *decoded pictures* are output from the *decoded picture buffer* (for the *decoded pictures* that are to be output from the *decoded picture buffer*), or the order in which the *access units* are considered to be output from the *decoded picture buffer* (for the *access units* the decoding of which results into *decoded pictures* that are to be output from the *decoded picture buffer*).
- F.3.40 partitioning scheme:** A *partitioning* of *layers* in an *OLS* into one or more *bitstream partitions* such that each *layer* in the *OLS* is included in exactly one *bitstream partition* of the partitioning scheme and each *bitstream partition* of the partitioning scheme contains one or more *layers*.
- F.3.41 partition unit:** A subset of *access unit* consisting of the *picture units* of the *layers* in a *bitstream partition*.
- F.3.42 picture order count (POC):** A variable that is associated with each *picture* and that, at any moment, uniquely identifies the associated *picture* among all *pictures* with the same value of *nuh_layer_id* in the *CVS*, and, when the associated *picture* is to be output from the *decoded picture buffer*, indicates the position of the associated *picture* in *output order* relative to the *output order* positions of the other *pictures* with the same value of *nuh_layer_id* in the same *CVS* that are to be output from the *decoded picture buffer*.
- F.3.43 picture order count (POC) resetting period:** A sequence of *access units* in *decoding order*, starting with an *access unit* with *poc_reset_idc* equal to 1 or 2 and a particular value of *poc_reset_period_id* and including all *access units* that either have the same value of *poc_reset_period_id* or have *poc_reset_idc* equal to 0.
- F.3.44 picture order count (POC) resetting picture:** A *picture* that is the first *picture*, in *decoding order*, of a *layer* of a *POC resetting period*.
- F.3.45 predicted layer:** A *layer* that is a *direct predicted layer* or an *indirect predicted layer* of another *layer*.
- F.3.46 primary picture:** A *picture* with *nuh_layer_id* value such that *AuxId[nuh_layer_id]* is equal to 0.
- F.3.47 primary picture layer:** A *layer* with a *nuh_layer_id* value such that *AuxId[nuh_layer_id]* is equal to 0.
- F.3.48 quality scalability:** A type of scalability, represented by a scalability dimension, that indicates that the *coded pictures* in all *layers* of a *bitstream* have the same values of *luma* width, *luma* height, *chroma* width and *chroma* height, but may have different fidelity level or different values of *luma* or *chroma* bit depths.
- F.3.49 reference layer:** A *layer* that is a *direct reference layer* or an *indirect reference layer* of another *layer*.
- F.3.50 reference picture list:** A list of *reference pictures* that is used for *inter prediction* or *inter-layer prediction* of a *P* or *B slice*.
- F.3.51 scalability dimension:** An indication of the type of *non-base layers* which may be present in a *CVS*.
- F.3.52 source picture for inter-layer prediction:** A *decoded picture* that either is, or is used in deriving, an *inter-layer reference picture* that is included in an *inter-layer reference picture set* of the current *picture*.
- F.3.53 spatial scalability:** A type of scalability, represented by a scalability dimension, that indicates that the *coded pictures* in different *layers* of a *bitstream* have different values of *luma* or *chroma* width or height.
- F.3.54 sub-bitstream extraction process:** A specified process by which *NAL units* in a *bitstream* that do not belong to a target set, determined by a target highest *TemporalId* and a target *layer identifier list*, are removed from the *bitstream*, with the output sub-bitstream consisting of the *NAL units* in the *bitstream* that belong to the target set.
- NOTE – Depending on whether the target layer identifier list includes *nuh_layer_id* equal to 0 and on the value of *vps_base_layer_internal_flag*, the sub-bitstream extraction process may refer to the process specified in clause 10, clause F.10.1 or clause F.10.3.
- F.3.55 target output layer set:** The *output layer set* for which the index is equal to *TargetOlsIdx*.
- F.3.56 trailing picture:** A *picture* that is in the same *layer* as the associated *IRAP picture* and follows the associated *IRAP picture* in *output order*.
- F.3.57 view:** A sequence of *pictures* associated with the same value of *ViewOrderIdx*.
- NOTE – A view typically represents a sequence of pictures captured by one camera.

F.4 Abbreviations

For the purpose of this annex, the following abbreviations apply in addition to the abbreviations in clause 4:

BPB	Bitstream Partition Buffer
HBPS	Hypothetical Bitstream Partition Scheduler
OLS	Output Layer Set

F.5 Conventions

The specifications in clause 5 and its subclauses apply.

F.6 Bitstream and picture formats, partitionings, scanning processes and neighbouring relationships

The specifications in clause 6 and its subclauses apply.

F.7 Syntax and semantics

F.7.1 Method of specifying syntax in tabular form

The specifications in clause 7.1 apply.

F.7.2 Specification of syntax functions, categories and descriptors

The specifications in clause 7.2 apply, with the following additions:

more_data_in_slice_segment_header_extension() is specified as follows:

- If (the current position in the slice_segment_header() syntax structure) – (the position immediately following slice_segment_header_extension_length) is less than (slice_segment_header_extension_length * 8), the return value of more_data_in_slice_segment_header_extension() is equal to TRUE.
- Otherwise, the return value of more_data_in_slice_segment_header_extension() is equal to FALSE.

F.7.3 Syntax in tabular form

F.7.3.1 NAL unit syntax

F.7.3.1.1 General NAL unit syntax

The specifications in clause 7.3.1.1 apply.

F.7.3.1.2 NAL unit header syntax

The specifications in clause 7.3.1.2 apply.

F.7.3.2 Raw byte sequence payloads and RBSP trailing bits syntax

F.7.3.2.1 Video parameter set RBSP

video_parameter_set_rbsp() {	Descriptor
vps_video_parameter_set_id	u(4)
vps_base_layer_internal_flag	u(1)
vps_base_layer_available_flag	u(1)
vps_max_layers_minus1	u(6)
vps_max_sub_layers_minus1	u(3)
vps_temporal_id_nesting_flag	u(1)
vps_reserved_0xffff_16bits	u(16)
profile_tier_level(1, vps_max_sub_layers_minus1)	
vps_sub_layer_ordering_info_present_flag	u(1)
for(i = (vps_sub_layer_ordering_info_present_flag ? 0 : vps_max_sub_layers_minus1); i <= vps_max_sub_layers_minus1; i++) {	
vps_max_dec_pic_buffering_minus1[i]	ue(v)
vps_max_num_reorder_pics[i]	ue(v)
vps_max_latency_increase_plus1[i]	ue(v)
}	
vps_max_layer_id	u(6)
vps_num_layer_sets_minus1	ue(v)
for(i = 1; i <= vps_num_layer_sets_minus1; i++)	

for(j = 0; j <= vps_max_layer_id; j++)	
layer_id_included_flag [i][j]	u(1)
vps_timing_info_present_flag	u(1)
if(vps_timing_info_present_flag) {	
vps_num_units_in_tick	u(32)
vps_time_scale	u(32)
vps_poc_proportional_to_timing_flag	u(1)
if(vps_poc_proportional_to_timing_flag)	
vps_num_ticks_poc_diff_one_minus1	ue(v)
vps_num_hrd_parameters	ue(v)
for(i = 0; i < vps_num_hrd_parameters; i++) {	
hrd_layer_set_idx [i]	ue(v)
if(i > 0)	
cprms_present_flag [i]	u(1)
hrd_parameters(cprms_present_flag[i], vps_max_sub_layers_minus1)	
}	
}	
vps_extension_flag	u(1)
if(vps_extension_flag) {	
while(!byte_aligned())	
vps_extension_alignment_bit_equal_to_one	u(1)
vps_extension()	
vps_extension2_flag	u(1)
if(vps_extension2_flag)	
while(more_rbsp_data())	
vps_extension_data_flag	u(1)
}	
rbsp_trailing_bits()	
}	

F.7.3.2.1.1 Video parameter set extension syntax

vps_extension() {	Descriptor
if(vps_max_layers_minus1 > 0 && vps_base_layer_internal_flag)	
profile_tier_level(0, vps_max_sub_layers_minus1)	
splitting_flag	u(1)
for(i = 0, NumScalabilityTypes = 0; i < 16; i++) {	
scalability_mask_flag [i]	u(1)
NumScalabilityTypes += scalability_mask_flag[i]	
}	
for(j = 0; j < (NumScalabilityTypes - splitting_flag); j++)	
dimension_id_len_minus1 [j]	u(3)
vps_nuh_layer_id_present_flag	u(1)
for(i = 1; i <= MaxLayersMinus1; i++) {	
if(vps_nuh_layer_id_present_flag)	
layer_id_in_nuh [i]	u(6)
if(!splitting_flag)	

for(j = 0; j < NumScalabilityTypes; j++)	
dimension_id[i][j]	u(v)
}	
view_id_len	u(4)
if(view_id_len > 0)	
for(i = 0; i < NumViews; i++)	
view_id_val[i]	u(v)
for(i = 1; i <= MaxLayersMinus1; i++)	
for(j = 0; j < i; j++)	
direct_dependency_flag[i][j]	u(1)
if(NumIndependentLayers > 1)	
num_add_layer_sets	ue(v)
for(i = 0; i < num_add_layer_sets; i++)	
for(j = 1; j < NumIndependentLayers; j++)	
highest_layer_idx_plus1[i][j]	u(v)
vps_sub_layers_max_minus1_present_flag	u(1)
if(vps_sub_layers_max_minus1_present_flag)	
for(i = 0; i <= MaxLayersMinus1; i++)	
sub_layers_vps_max_minus1[i]	u(3)
max_tid_ref_present_flag	u(1)
if(max_tid_ref_present_flag)	
for(i = 0; i < MaxLayersMinus1; i++)	
for(j = i + 1; j <= MaxLayersMinus1; j++)	
if(direct_dependency_flag[j][i])	
max_tid_il_ref_pics_plus1[i][j]	u(3)
default_ref_layers_active_flag	u(1)
vps_num_profile_tier_level_minus1	ue(v)
for(i = vps_base_layer_internal_flag ? 2 : 1; i <= vps_num_profile_tier_level_minus1; i++) {	
vps_profile_present_flag[i]	u(1)
profile_tier_level(vps_profile_present_flag[i], vps_max_sub_layers_minus1)	
}	
if(NumLayerSets > 1) {	
num_add_olss	ue(v)
default_output_layer_idc	u(2)
}	
NumOutputLayerSets = num_add_olss + NumLayerSets	
for(i = 1; i < NumOutputLayerSets; i++) {	
if(NumLayerSets > 2 && i >= NumLayerSets)	
layer_set_idx_for_ols_minus1[i]	u(v)
if(i > vps_num_layer_sets_minus1 defaultOutputLayerIdc == 2)	
for(j = 0; j < NumLayersInIdList[OlsIdxToLsIdx[i]]; j++)	
output_layer_flag[i][j]	u(1)
for(j = 0; j < NumLayersInIdList[OlsIdxToLsIdx[i]]; j++)	
if(NecessaryLayerFlag[i][j] && vps_num_profile_tier_level_minus1 > 0)	
profile_tier_level_idx[i][j]	u(v)
if(NumOutputLayersInOutputLayerSet[i] == 1 && NumDirectRefLayers[OlsHighestOutputLayerId[i]] > 0)	
alt_output_layer_flag[i]	u(1)

}	
vps_num_rep_formats_minus1	ue(v)
for(i = 0; i <= vps_num_rep_formats_minus1; i++)	
rep_format()	
if(vps_num_rep_formats_minus1 > 0)	
rep_format_idx_present_flag	u(1)
if(rep_format_idx_present_flag)	
for(i = vps_base_layer_internal_flag ? 1 : 0; i <= MaxLayersMinus1; i++)	
vps_rep_format_idx[i]	u(v)
max_one_active_ref_layer_flag	u(1)
vps_poc_lsb_aligned_flag	u(1)
for(i = 1; i <= MaxLayersMinus1; i++)	
if(NumDirectRefLayers[layer_id_in_nuh[i]] == 0)	
poc_lsb_not_present_flag[i]	u(1)
dpb_size()	
direct_dep_type_len_minus2	ue(v)
direct_dependency_all_layers_flag	u(1)
if(direct_dependency_all_layers_flag)	
direct_dependency_all_layers_type	u(v)
else {	
for(i = vps_base_layer_internal_flag ? 1 : 2; i <= MaxLayersMinus1; i++)	
for(j = vps_base_layer_internal_flag ? 0 : 1; j < i; j++)	
if(direct_dependency_flag[i][j])	
direct_dependency_type[i][j]	u(v)
}	
vps_non_vui_extension_length	ue(v)
for(i = 1; i <= vps_non_vui_extension_length; i++)	
vps_non_vui_extension_data_byte	u(8)
vps_vui_present_flag	u(1)
if(vps_vui_present_flag) {	
while(!byte_aligned())	
vps_vui_alignment_bit_equal_to_one	u(1)
vps_vui()	
}	
}	

F.7.3.2.1.2 Representation format syntax

rep_format() {	Descriptor
pic_width_vps_in_luma_samples	u(16)
pic_height_vps_in_luma_samples	u(16)
chroma_and_bit_depth_vps_present_flag	u(1)
if(chroma_and_bit_depth_vps_present_flag) {	
chroma_format_vps_idc	u(2)
if(chroma_format_vps_idc == 3)	
separate_colour_plane_vps_flag	u(1)
bit_depth_vps_luma_minus8	u(4)
bit_depth_vps_chroma_minus8	u(4)
}	
conformance_window_vps_flag	u(1)
if(conformance_window_vps_flag) {	
conf_win_vps_left_offset	ue(v)
conf_win_vps_right_offset	ue(v)
conf_win_vps_top_offset	ue(v)
conf_win_vps_bottom_offset	ue(v)
}	
}	

F.7.3.2.1.3 DPB size syntax

dpb_size() {	Descriptor
for(i = 1; i < NumOutputLayerSets; i++) {	
currLsIdx = OlsIdxToLsIdx[i]	
sub_layer_flag_info_present_flag[i]	u(1)
for(j = 0; j <= MaxSubLayersInLayerSetMinus1[currLsIdx]; j++) {	
if(j > 0 && sub_layer_flag_info_present_flag[i])	
sub_layer_dpb_info_present_flag[i][j]	u(1)
if(sub_layer_dpb_info_present_flag[i][j]) {	
for(k = 0; k < NumLayersInIdList[currLsIdx]; k++)	
if(NecessaryLayerFlag[i][k] && (vps_base_layer_internal_flag (LayerSetLayerIdList[currLsIdx][k] != 0)))	
max_vps_dec_pic_buffering_minus1[i][k][j]	ue(v)
max_vps_num_reorder_pics[i][j]	ue(v)
max_vps_latency_increase_plus1[i][j]	ue(v)
}	
}	
}	
}	

F.7.3.2.1.4 VPS VUI syntax

vps_vui() {	Descriptor
cross_layer_pic_type_aligned_flag	u(1)
if(!cross_layer_pic_type_aligned_flag)	
cross_layer_irap_aligned_flag	u(1)
if(cross_layer_irap_aligned_flag)	
all_layers_idr_aligned_flag	u(1)
bit_rate_present_vps_flag	u(1)
pic_rate_present_vps_flag	u(1)
if(bit_rate_present_vps_flag pic_rate_present_vps_flag)	
for(i = vps_base_layer_internal_flag ? 0 : 1; i < NumLayerSets; i++)	
for(j = 0; j <= MaxSubLayersInLayerSetMinus1[i]; j++) {	
if(bit_rate_present_vps_flag)	
bit_rate_present_flag[i][j]	u(1)
if(pic_rate_present_vps_flag)	
pic_rate_present_flag[i][j]	u(1)
if(bit_rate_present_flag[i][j]) {	
avg_bit_rate[i][j]	u(16)
max_bit_rate[i][j]	u(16)
}	
if(pic_rate_present_flag[i][j]) {	
constant_pic_rate_idc[i][j]	u(2)
avg_pic_rate[i][j]	u(16)
}	
}	
video_signal_info_idx_present_flag	u(1)
if(video_signal_info_idx_present_flag)	
vps_num_video_signal_info_minus1	u(4)
for(i = 0; i <= vps_num_video_signal_info_minus1; i++)	
video_signal_info()	
if(video_signal_info_idx_present_flag && vps_num_video_signal_info_minus1 > 0)	
for(i = vps_base_layer_internal_flag ? 0 : 1; i <= MaxLayersMinus1; i++)	
vps_video_signal_info_idx[i]	u(4)
tiles_not_in_use_flag	u(1)
if(!tiles_not_in_use_flag) {	
for(i = vps_base_layer_internal_flag ? 0 : 1; i <= MaxLayersMinus1; i++) {	
tiles_in_use_flag[i]	u(1)
if(tiles_in_use_flag[i])	
loop_filter_not_across_tiles_flag[i]	u(1)
}	
for(i = vps_base_layer_internal_flag ? 1 : 2; i <= MaxLayersMinus1; i++)	
for(j = 0; j < NumDirectRefLayers[layer_id_in_nuh[i]]; j++) {	
layerIdx = LayerIdxInVps[IdDirectRefLayer[layer_id_in_nuh[i]][j]]	
if(tiles_in_use_flag[i] && tiles_in_use_flag[layerIdx])	
tile_boundaries_aligned_flag[i][j]	u(1)
}	
}	
wpp_not_in_use_flag	u(1)

if(!wpp_not_in_use_flag)	
for(i = vps_base_layer_internal_flag ? 0 : 1; i <= MaxLayersMinus1; i++)	
wpp_in_use_flag[i]	u(1)
single_layer_for_non_irap_flag	u(1)
higher_layer_irap_skip_flag	u(1)
ilp_restricted_ref_layers_flag	u(1)
if(ilp_restricted_ref_layers_flag)	
for(i = 1; i <= MaxLayersMinus1; i++)	
for(j = 0; j < NumDirectRefLayers[layer_id_in_nuh[i]]; j++)	
if(vps_base_layer_internal_flag IdDirectRefLayer[layer_id_in_nuh[i]][j] > 0) {	
min_spatial_segment_offset_plus1[i][j]	ue(v)
if(min_spatial_segment_offset_plus1[i][j] > 0) {	
ctu_based_offset_enabled_flag[i][j]	u(1)
if(ctu_based_offset_enabled_flag[i][j])	
min_horizontal_ctu_offset_plus1[i][j]	ue(v)
}	
}	
vps_vui_bsp_hrd_present_flag	u(1)
if(vps_vui_bsp_hrd_present_flag)	
vps_vui_bsp_hrd_params()	
for(i = 1; i <= MaxLayersMinus1; i++)	
if(NumDirectRefLayers[layer_id_in_nuh[i]] == 0)	
base_layer_parameter_set_compatibility_flag[i]	u(1)
}	

F.7.3.2.1.5 Video signal info syntax

video_signal_info() {	Descriptor
video_vps_format	u(3)
video_full_range_vps_flag	u(1)
colour_primaries_vps	u(8)
transfer_characteristics_vps	u(8)
matrix_coeffs_vps	u(8)
}	

F.7.3.2.1.6 VPS VUI bitstream partition HRD parameters syntax

vps_vui_bsp_hrd_params() {	Descriptor
vps_num_add_hrd_params	ue(v)
for(i = vps_num_hrd_parameters; i < vps_num_hrd_parameters + vps_num_add_hrd_params; i++) {	
if(i > 0)	
cprms_add_present_flag[i]	u(1)
num_sub_layer_hrd_minus1[i]	ue(v)
hrd_parameters(cprms_add_present_flag[i], num_sub_layer_hrd_minus1[i])	
}	
if(vps_num_hrd_parameters + vps_num_add_hrd_params > 0)	
for(h = 1; h < NumOutputLayerSets; h++) {	
num_signalled_partitioning_schemes[h]	ue(v)
for(j = 1; j < num_signalled_partitioning_schemes[h] + 1; j++) {	
num_partitions_in_scheme_minus1[h][j]	ue(v)
for(k = 0; k <= num_partitions_in_scheme_minus1[h][j]; k++)	
for(r = 0; r < NumLayersInIdList[OlsIdxToLsIdx[h]]; r++)	
layer_included_in_partition_flag[h][j][k][r]	u(1)
}	
for(i = 0; i < num_signalled_partitioning_schemes[h] + 1; i++)	
for(t = 0; t <= MaxSubLayersInLayerSetMinus1[OlsIdxToLsIdx[h]]; t++) {	
num_bsp_schedules_minus1[h][i][t]	ue(v)
for(j = 0; j <= num_bsp_schedules_minus1[h][i][t]; j++)	
for(k = 0; k <= num_partitions_in_scheme_minus1[h][i]; k++) {	
if(vps_num_hrd_parameters + vps_num_add_hrd_params > 1)	
bsp_hrd_idx[h][i][t][j][k]	u(v)
bsp_sched_idx[h][i][t][j][k]	ue(v)
}	
}	
}	
}	

F.7.3.2.2 Sequence parameter set RBSP syntax

F.7.3.2.2.1 General sequence parameter set RBSP syntax

seq_parameter_set_rbsp() {	Descriptor
sps_video_parameter_set_id	u(4)
if(nuh_layer_id == 0)	
sps_max_sub_layers_minus1	u(3)
else	
sps_ext_or_max_sub_layers_minus1	u(3)
MultiLayerExtSpsFlag = (nuh_layer_id != 0 && sps_ext_or_max_sub_layers_minus1 == 7)	
if(!MultiLayerExtSpsFlag) {	
sps_temporal_id_nesting_flag	u(1)
profile_tier_level(1, sps_max_sub_layers_minus1)	
}	

sps_seq_parameter_set_id	ue(v)
if(MultiLayerExtSpsFlag) {	
update_rep_format_flag	u(1)
if(update_rep_format_flag)	
sps_rep_format_idx	u(8)
} else {	
chroma_format_idc	ue(v)
if(chroma_format_idc == 3)	
separate_colour_plane_flag	u(1)
pic_width_in_luma_samples	ue(v)
pic_height_in_luma_samples	ue(v)
conformance_window_flag	u(1)
if(conformance_window_flag) {	
conf_win_left_offset	ue(v)
conf_win_right_offset	ue(v)
conf_win_top_offset	ue(v)
conf_win_bottom_offset	ue(v)
}	
bit_depth_luma_minus8	ue(v)
bit_depth_chroma_minus8	ue(v)
}	
log2_max_pic_order_cnt_lsb_minus4	ue(v)
if(!MultiLayerExtSpsFlag) {	
sps_sub_layer_ordering_info_present_flag	u(1)
for(i = (sps_sub_layer_ordering_info_present_flag ? 0 : sps_max_sub_layers_minus1);	
i <= sps_max_sub_layers_minus1; i++) {	
sps_max_dec_pic_buffering_minus1[i]	ue(v)
sps_max_num_reorder_pics[i]	ue(v)
sps_max_latency_increase_plus1[i]	ue(v)
}	
}	
log2_min_luma_coding_block_size_minus3	ue(v)
log2_diff_max_min_luma_coding_block_size	ue(v)
log2_min_luma_transform_block_size_minus2	ue(v)
log2_diff_max_min_luma_transform_block_size	ue(v)
max_transform_hierarchy_depth_inter	ue(v)
max_transform_hierarchy_depth_intra	ue(v)
scaling_list_enabled_flag	u(1)
if(scaling_list_enabled_flag) {	
if(MultiLayerExtSpsFlag)	
sps_infer_scaling_list_flag	u(1)
if(sps_infer_scaling_list_flag)	
sps_scaling_list_ref_layer_id	u(6)
else {	
sps_scaling_list_data_present_flag	u(1)
if(sps_scaling_list_data_present_flag)	
scaling_list_data()	
}	
}	

amp_enabled_flag	u(1)
sample_adaptive_offset_enabled_flag	u(1)
pcm_enabled_flag	u(1)
if(pcm_enabled_flag) {	
pcm_sample_bit_depth_luma_minus1	u(4)
pcm_sample_bit_depth_chroma_minus1	u(4)
log2_min_pcm_luma_coding_block_size_minus3	ue(v)
log2_diff_max_min_pcm_luma_coding_block_size	ue(v)
pcm_loop_filter_disabled_flag	u(1)
}	
num_short_term_ref_pic_sets	ue(v)
for(i = 0; i < num_short_term_ref_pic_sets; i++)	
st_ref_pic_set(i)	
long_term_ref_pics_present_flag	u(1)
if(long_term_ref_pics_present_flag) {	
num_long_term_ref_pics_sps	ue(v)
for(i = 0; i < num_long_term_ref_pics_sps; i++) {	
lt_ref_pic_poc_lsb_sps[i]	u(v)
used_by_curr_pic_lt_sps_flag[i]	u(1)
}	
}	
sps_temporal_mvp_enabled_flag	u(1)
strong_intra_smoothing_enabled_flag	u(1)
vui_parameters_present_flag	u(1)
if(vui_parameters_present_flag)	
vui_parameters()	
sps_extension_present_flag	u(1)
if(sps_extension_present_flag) {	
sps_range_extension_flag	u(1)
sps_multilayer_extension_flag	u(1)
sps_3d_extension_flag	u(1)
sps_scc_extension_flag	u(1)
sps_extension_4bits	u(5)
}	
if(sps_range_extension_flag)	
sps_range_extension()	
if(sps_multilayer_extension_flag)	
sps_multilayer_extension()	
if(sps_3d_extension_flag)	
sps_3d_extension() /* specified in Annex I */	
if(sps_scc_extension_flag)	
sps_scc_extension()	
if(sps_extension_4bits)	
while(more_rbsp_data())	
sps_extension_data_flag	u(1)
rbsp_trailing_bits()	
}	

F.7.3.2.2.2 Sequence parameter set range extension syntax

The specifications in clause 7.3.2.2.2 apply.

F.7.3.2.2.3 Sequence parameter set screen content coding extension syntax

The specifications in clause 7.3.2.2.3 apply.

F.7.3.2.2.4 Sequence parameter set multilayer extension syntax

	Descriptor
sps_multilayer_extension() {	
inter_view_mv_vert_constraint_flag	u(1)
}	

F.7.3.2.3 Picture parameter set RBSP syntax

F.7.3.2.3.1 General picture parameter set RBSP syntax

The specifications in clause 7.3.2.3.1 apply.

F.7.3.2.3.2 Picture parameter set range extension syntax

The specifications in clause 7.3.2.3.2 apply.

F.7.3.2.3.3 Picture parameter set screen content coding extension syntax

The specifications in clause 7.3.2.3.3 apply.

F.7.3.2.3.4 Picture parameter set multilayer extension syntax

pps_multilayer_extension() {	Descriptor
poc_reset_info_present_flag	u(1)
pps_infer_scaling_list_flag	u(1)
if(pps_infer_scaling_list_flag)	
pps_scaling_list_ref_layer_id	u(6)
num_ref_loc_offsets	ue(v)
for(i = 0; i < num_ref_loc_offsets; i++) {	
ref_loc_offset_layer_id[i]	u(6)
scaled_ref_layer_offset_present_flag[i]	u(1)
if(scaled_ref_layer_offset_present_flag[i]) {	
scaled_ref_layer_left_offset[ref_loc_offset_layer_id[i]]	se(v)
scaled_ref_layer_top_offset[ref_loc_offset_layer_id[i]]	se(v)
scaled_ref_layer_right_offset[ref_loc_offset_layer_id[i]]	se(v)
scaled_ref_layer_bottom_offset[ref_loc_offset_layer_id[i]]	se(v)
}	
ref_region_offset_present_flag[i]	u(1)
if(ref_region_offset_present_flag[i]) {	
ref_region_left_offset[ref_loc_offset_layer_id[i]]	se(v)
ref_region_top_offset[ref_loc_offset_layer_id[i]]	se(v)
ref_region_right_offset[ref_loc_offset_layer_id[i]]	se(v)
ref_region_bottom_offset[ref_loc_offset_layer_id[i]]	se(v)
}	
resample_phase_set_present_flag[i]	u(1)
if(resample_phase_set_present_flag[i]) {	
phase_hor_luma[ref_loc_offset_layer_id[i]]	ue(v)
phase_ver_luma[ref_loc_offset_layer_id[i]]	ue(v)
phase_hor_chroma_plus8[ref_loc_offset_layer_id[i]]	ue(v)
phase_ver_chroma_plus8[ref_loc_offset_layer_id[i]]	ue(v)
}	
}	
colour_mapping_enabled_flag	u(1)
if(colour_mapping_enabled_flag)	
colour_mapping_table()	
}	

F.7.3.2.3.5 General colour mapping table syntax

colour_mapping_table() {	Descriptor
num_cm_ref_layers_minus1	ue(v)
for(i = 0; i <= num_cm_ref_layers_minus1; i++)	
cm_ref_layer_id[i]	u(6)
cm_octant_depth	u(2)
cm_y_part_num_log2	u(2)
luma_bit_depth_cm_input_minus8	ue(v)
chroma_bit_depth_cm_input_minus8	ue(v)
luma_bit_depth_cm_output_minus8	ue(v)
chroma_bit_depth_cm_output_minus8	ue(v)
cm_res_quant_bits	u(2)
cm_delta_flc_bits_minus1	u(2)
if(cm_octant_depth == 1) {	
cm_adapt_threshold_u_delta	se(v)
cm_adapt_threshold_v_delta	se(v)
}	
colour_mapping_octants(0, 0, 0, 0, 1 << cm_octant_depth)	
}	

F.7.3.2.3.6 Colour mapping octants syntax

colour_mapping_octants(inpDepth, idxY, idxCb, idxCr, inpLength) {	Descriptor
if(inpDepth < cm_octant_depth)	
split_octant_flag	u(1)
if(split_octant_flag)	
for(k = 0; k < 2; k++)	
for(m = 0; m < 2; m++)	
for(n = 0; n < 2; n++)	
colour_mapping_octants(inpDepth + 1, idxY + PartNumY * k * inpLength / 2, idxCb + m * inpLength / 2, idxCr + n * inpLength / 2, inpLength / 2)	
else	
for(i = 0; i < PartNumY; i++) {	
idxShiftY = idxY + (i << (cm_octant_depth - inpDepth))	
for(j = 0; j < 4; j++) {	
coded_res_flag[idxShiftY][idxCb][idxCr][j]	u(1)
if(coded_res_flag[idxShiftY][idxCb][idxCr][j])	
for(c = 0; c < 3; c++) {	
res_coeff_q[idxShiftY][idxCb][idxCr][j][c]	ue(v)
res_coeff_r[idxShiftY][idxCb][idxCr][j][c]	u(v)
if(res_coeff_q[idxShiftY][idxCb][idxCr][j][c] res_coeff_r[idxShiftY][idxCb][idxCr][j][c])	
res_coeff_s[idxShiftY][idxCb][idxCr][j][c]	u(1)
}	
}	
}	
}	

F.7.3.2.4 Supplemental enhancement information RBSP syntax

The specifications in clause 7.3.2.4 apply.

F.7.3.2.5 Access unit delimiter RBSP syntax

The specifications in clause 7.3.2.5 apply.

F.7.3.2.6 End of sequence RBSP syntax

The specifications in clause 7.3.2.6 apply.

F.7.3.2.7 End of bitstream RBSP syntax

The specifications in clause 7.3.2.7 apply.

F.7.3.2.8 Filler data RBSP syntax

The specifications in clause 7.3.2.8 apply.

F.7.3.2.9 Slice segment layer RBSP syntax

The specifications in clause 7.3.2.9 apply.

F.7.3.2.10 RBSP slice segment trailing bits syntax

The specifications in clause 7.3.2.10 apply.

F.7.3.2.11 RBSP trailing bits syntax

The specifications in clause 7.3.2.11 apply.

F.7.3.2.12 Byte alignment syntax

The specifications in clause 7.3.2.12 apply.

F.7.3.3 Profile, tier and level syntax

The specifications in clause 7.3.3 apply.

F.7.3.4 Scaling list data syntax

The specifications in clause 7.3.4 apply.

F.7.3.5 Supplemental enhancement information message syntax

The specifications in clause 7.3.5 apply.

F.7.3.6 Slice segment header syntax

F.7.3.6.1 General slice segment header syntax

<code>slice_segment_header() {</code>	Descriptor
<code>first_slice_segment_in_pic_flag</code>	<code>u(1)</code>
<code>if(nal_unit_type >= BLA_W_LP && nal_unit_type <= RSV_IRAP_VCL23)</code>	
<code>no_output_of_prior_pics_flag</code>	<code>u(1)</code>
<code>slice_pic_parameter_set_id</code>	<code>ue(v)</code>
<code>if(!first_slice_segment_in_pic_flag) {</code>	
<code>if(dependent_slice_segments_enabled_flag)</code>	
<code>dependent_slice_segment_flag</code>	<code>u(1)</code>
<code>slice_segment_address</code>	<code>u(v)</code>
<code>}</code>	
<code>if(!dependent_slice_segment_flag) {</code>	
<code>i = 0</code>	
<code>if(num_extra_slice_header_bits > i) {</code>	

i++	
discardable_flag	u(1)
}	
if(num_extra_slice_header_bits > i) {	
i++	
cross_layer_bla_flag	u(1)
}	
for(; i < num_extra_slice_header_bits; i++)	
slice_reserved_flag[i]	u(1)
slice_type	ue(v)
if(output_flag_present_flag)	
pic_output_flag	u(1)
if(separate_colour_plane_flag == 1)	
colour_plane_id	u(2)
if((nuh_layer_id > 0 && !poc_lsb_not_present_flag[LayerIdxInVps[nuh_layer_id]]) (nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP))	
slice_pic_order_cnt_lsb	u(v)
if(nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP) {	
short_term_ref_pic_set_sps_flag	u(1)
if(!short_term_ref_pic_set_sps_flag)	
st_ref_pic_set(num_short_term_ref_pic_sets)	
else if(num_short_term_ref_pic_sets > 1)	
short_term_ref_pic_set_idx	u(v)
if(long_term_ref_pics_present_flag) {	
if(num_long_term_ref_pics_sps > 0)	
num_long_term_sps	ue(v)
num_long_term_pics	ue(v)
for(i = 0; i < num_long_term_sps + num_long_term_pics; i++) {	
if(i < num_long_term_sps) {	
if(num_long_term_ref_pics_sps > 1)	
lt_idx_sps[i]	u(v)
} else {	
poc_lsb_lt[i]	u(v)
used_by_curr_pic_lt_flag[i]	u(1)
}	
delta_poc_msb_present_flag[i]	u(1)
if(delta_poc_msb_present_flag[i])	
delta_poc_msb_cycle_lt[i]	ue(v)
}	
}	
if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)
}	
if(nuh_layer_id > 0 && !default_ref_layers_active_flag && NumDirectRefLayers[nuh_layer_id] > 0) {	
inter_layer_pred_enabled_flag	u(1)
if(inter_layer_pred_enabled_flag && NumDirectRefLayers[nuh_layer_id] > 1) {	
if(!max_one_active_ref_layer_flag)	

num_inter_layer_ref_pics_minus1	u(v)
if(NumActiveRefLayerPics != NumDirectRefLayers[nuh_layer_id])	
for(i = 0; i < NumActiveRefLayerPics; i++)	
inter_layer_pred_layer_idc[i]	u(v)
}	
}	
if(sample_adaptive_offset_enabled_flag) {	
slice_sao_luma_flag	u(1)
if(ChromaArrayType != 0)	
slice_sao_chroma_flag	u(1)
}	
if(slice_type == P slice_type == B) {	
num_ref_idx_active_override_flag	u(1)
if(num_ref_idx_active_override_flag) {	
num_ref_idx_l0_active_minus1	ue(v)
if(slice_type == B)	
num_ref_idx_l1_active_minus1	ue(v)
}	
if(lists_modification_present_flag && NumPicTotalCurr > 1)	
ref_pic_lists_modification()	
if(slice_type == B)	
mvd_l1_zero_flag	u(1)
if(cabac_init_present_flag)	
cabac_init_flag	u(1)
if(slice_temporal_mvp_enabled_flag) {	
if(slice_type == B)	
collocated_from_l0_flag	u(1)
if((collocated_from_l0_flag && num_ref_idx_l0_active_minus1 > 0) (!collocated_from_l0_flag && num_ref_idx_l1_active_minus1 > 0))	
collocated_ref_idx	ue(v)
}	
if((weighted_pred_flag && slice_type == P) (weighted_bipred_flag && slice_type == B))	
pred_weight_table()	
five_minus_max_num_merge_cand	ue(v)
}	
slice_qp_delta	se(v)
if(pps_slice_chroma_qp_offsets_present_flag) {	
slice_cb_qp_offset	se(v)
slice_cr_qp_offset	se(v)
}	
if(chroma_qp_offset_list_enabled_flag)	
cu_chroma_qp_offset_enabled_flag	u(1)
if(deblocking_filter_override_enabled_flag)	
deblocking_filter_override_flag	u(1)
if(deblocking_filter_override_flag) {	
slice_deblocking_filter_disabled_flag	u(1)
if(!slice_deblocking_filter_disabled_flag) {	
slice_beta_offset_div2	se(v)

slice_tc_offset_div2	se(v)
}	
}	
if(pps_loop_filter_across_slices_enabled_flag && (slice_sao_luma_flag slice_sao_chroma_flag !slice_deblocking_filter_disabled_flag))	
slice_loop_filter_across_slices_enabled_flag	u(1)
}	
if(tiles_enabled_flag entropy_coding_sync_enabled_flag) {	
num_entry_point_offsets	ue(v)
if(num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset_minus1[i]	u(v)
}	
}	
if(slice_segment_header_extension_present_flag) {	
slice_segment_header_extension_length	ue(v)
if(poc_reset_info_present_flag)	
poc_reset_idc	u(2)
if(poc_reset_idc != 0)	
poc_reset_period_id	u(6)
if(poc_reset_idc == 3) {	
full_poc_reset_flag	u(1)
poc_lsb_val	u(v)
}	
if(!PocMsbValRequiredFlag && vps_poc_lsb_aligned_flag)	
poc_msb_cycle_val_present_flag	u(1)
if(poc_msb_cycle_val_present_flag)	
poc_msb_cycle_val	ue(v)
while(more_data_in_slice_segment_header_extension())	
slice_segment_header_extension_data_bit	u(1)
}	
byte_alignment()	
}	

F.7.3.6.2 Reference picture list modification syntax

The specifications in clause 7.3.6.2 apply.

F.7.3.6.3 Weighted prediction parameters syntax

The specifications in clause 7.3.6.3 apply.

F.7.3.7 Short-term reference picture set syntax

The specifications in clause 7.3.7 apply.

F.7.3.8 Slice segment data syntax

F.7.3.8.1 General slice segment data syntax

The specifications in clause 7.3.8.1 apply.

F.7.3.8.2 Coding tree unit syntax

The specifications in clause 7.3.8.2 apply.

F.7.3.8.3 Sample adaptive offset syntax

The specifications in clause 7.3.8.3 apply.

F.7.3.8.4 Coding quadtree syntax

The specifications in clause 7.3.8.4 apply.

F.7.3.8.5 Coding unit syntax

The specifications in clause 7.3.8.5 apply.

F.7.3.8.6 Prediction unit syntax

The specifications in clause 7.3.8.6 apply.

F.7.3.8.7 PCM sample syntax

The specifications in clause 7.3.8.7 apply.

F.7.3.8.8 Transform tree syntax

The specifications in clause 7.3.8.8 apply.

F.7.3.8.9 Motion vector difference syntax

The specifications in clause 7.3.8.9 apply.

F.7.3.8.10 Transform unit syntax

The specifications in clause 7.3.8.10 apply.

F.7.3.8.11 Residual coding syntax

The specifications in clause 7.3.8.11 apply.

F.7.3.8.12 Cross-component prediction syntax

The specifications in clause 7.3.8.12 apply.

F.7.3.8.13 Palette mode syntax

The specifications in clause 7.3.8.13 apply.

F.7.3.8.14 Delta QP syntax

The specifications in clause 7.3.8.14 apply.

F.7.3.8.15 Chroma QP offset syntax

The specifications in clause 7.3.8.15 apply.

F.7.4 Semantics

F.7.4.1 General

F.7.4.2 NAL unit semantics

F.7.4.2.1 General NAL unit semantics

The specifications in clause 7.4.2.1 apply.

F.7.4.2.2 NAL unit header semantics

The specifications in clause 7.4.2.2 apply with the replacement of references to Annex C with references to clause F.13 and with the following additions:

The variable CraOrBlaPicFlag is derived as follows:

$$\text{CraOrBlaPicFlag} = (\text{nal_unit_type} == \text{BLA_W_LP} \mid \mid \text{nal_unit_type} == \text{BLA_N_LP} \mid \mid \text{nal_unit_type} == \text{BLA_W_RADL} \mid \mid \text{nal_unit_type} == \text{CRA_NUT}) \quad (\text{F-1})$$

NOTE 1 – When a picture picA that is a CRA picture and belongs to a layer with nuh_layer_id equal to layerId is present in a bitstream and pictures belonging to the layer with nuh_layer_id equal to layerId and preceding, in decoding order, the picture picA are dropped due to layer down-switching followed by layer up-switching, the RASL pictures associated with the picture picA, if any, may have some reference pictures that may not be available for reference unless one of the following conditions is true:

- The access unit auA containing the picture picA is an IRAP access unit, and the picture with nuh_layer_id equal to SmallestLayerId in the access unit auA, if any, has NoCrasOutputFlag equal to 1.
- The value of HandleCraAsBlaFlag is equal to 1 for the CRA picture picA.

A NAL unit with nal_unit_type equal to EOS_NUT and with a particular nuh_layer_id value shall not be present in an access unit, unless a coded picture with that particular nuh_layer_id value is present in the same access unit.

NOTE 2 – Let indepLayerId be nuh_layer_id value of any independent non-base layer with nuh_layer_id greater than SmallestLayerId. Let firstPic be the first picture, in decoding order, among the pictures with nuh_layer_id equal to indepLayerId or IdPredictedLayer[indepLayerId][i] for any value of i in the range of 0 to NumPredictedLayers[indepLayerId] – 1, inclusive, that follows an IRAP picture with NoCrasOutputFlag equal to 1. firstPic has to be an IRAP picture with nuh_layer_id equal to indepLayerId and with LayerResetFlag equal to 1.

When nal_unit_type is equal to AUD_NUT, the value of nuh_layer_id shall be equal to the minimum of the nuh_layer_id values of all VCL NAL units in the access unit.

F.7.4.2.3 Encapsulation of an SODB within an RBSP (informative)

The specifications in clause 7.4.2.3 apply.

F.7.4.2.4 Order of NAL units and association to coded pictures, access units and coded video sequences

F.7.4.2.4.1 General

The specifications in clause 7.4.2.4.1 apply.

F.7.4.2.4.2 Order of VPS, SPS and PPS RBSPs and their activation

The specifications in clause 7.4.2.4.2 apply with the following additions:

A PPS RBSP includes parameters that can be referred to by the coded slice segment NAL units of one or more coded pictures. Each PPS RBSP is initially considered not active for any layer at the start of the operation of the decoding process. At most one PPS RBSP is considered active for each layer at any given moment during the operation of the decoding process and the activation of any particular PPS RBSP for a particular layer results in the deactivation of the previously-active PPS RBSP for the particular layer (if any).

One PPS RBSP may be the active PPS RBSP for more than one layer. When not explicitly specified, the layer a PPS RBSP is active for is inferred to be the current layer in the context where the active PPS RBSP is referred to.

When a PPS RBSP (with a particular value of pps_pic_parameter_set_id) is not active for a particular layer with nuh_layer_id activatingLayerId and it is referred to by a coded slice segment NAL unit (using a value of slice_pic_parameter_set_id equal to the pps_pic_parameter_set_id value) of the particular layer, it is activated for the particular layer. This PPS RBSP is called the active PPS RBSP for the particular layer until it is deactivated by the activation of another PPS RBSP for the particular layer. A PPS RBSP, with that particular value of pps_pic_parameter_set_id, shall be available to the decoding process prior to its activation, included in at least one access unit with TemporalId less than or equal to the TemporalId of the PPS NAL unit or provided through external means, and the PPS NAL unit containing the PPS RBSP shall have nuh_layer_id equal to 0, activatingLayerId, or IdRefLayer[activatingLayerId][i] with any value of i in the range of 0 to NumRefLayers[activatingLayerId] – 1, inclusive.

NOTE 1 – All PPSs, regardless of their values of nuh_layer_id or TemporalId, share the same value space for pps_pic_parameter_set_id. In other words, a PPS with nuh_layer_id equal to X, TemporalId equal to Y and pps_pic_parameter_set_id equal to A would update the previously received PPS that has pps_pic_parameter_set_id equal to A and that has nuh_layer_id not equal to X or TemporalId not equal to Y.

An SPS RBSP includes parameters that can be referred to by one or more PPS RBSPs or one or more SEI NAL units containing an active parameter sets SEI message. Each SPS RBSP is initially considered not active for any layer at the start of the operation of the decoding process. At most one SPS RBSP is considered active for each layer at any given moment during the operation of the decoding process and the activation of any particular SPS RBSP for a particular layer results in the deactivation of the previously-active SPS RBSP for that particular layer (if any).

One SPS RBSP may be the active SPS RBSP for more than one layer. When not explicitly specified, the layer an SPS RBSP is active for is inferred to be the current layer in the context where the active SPS RBSP is referred to.

When an SPS RBSP (with a particular value of `sps_seq_parameter_set_id`) is not already active for a particular non-base layer with `nuh_layer_id` `nuhLayerId` and it is referred to by activation of a PPS RBSP (in which `pps_seq_parameter_set_id` is equal to the `sps_seq_parameter_set_id` value), or is referred to by an SEI NAL unit containing an active parameter sets SEI message (in which `active_seq_parameter_set_id[layer_sps_idx[LayerIdxInVps[nuhLayerId]]]` is equal to the `sps_seq_parameter_set_id` value), it is activated for the particular non-base layer. This SPS RBSP is called the active SPS RBSP for the particular non-base layer until it is deactivated by the activation of another SPS RBSP for the particular non-base layer. An SPS RBSP, with the particular value of `sps_seq_parameter_set_id`, shall be available to the decoding process prior to its activation, included in at least one access unit with `TemporalId` equal to 0 or provided through external means, and the SPS NAL unit containing the SPS RBSP shall have `nuh_layer_id` equal to 0, `nuhLayerId`, or `IdRefLayer[nuhLayerId][i]` with any value of `i` in the range of 0 to `NumRefLayers[nuhLayerId] - 1`, inclusive. An activated SPS RBSP for a particular layer shall remain active for the entire CLVS of that particular layer.

The contents of the `hrd_parameters()` syntax structure shall remain unchanged within a sequence of activated SPS RBSPs, in their activation order, from any activated SPS RBSP until the end of the bitstream or up to but excluding an SPS RBSP that is activated within the next access unit in which at least one of the following conditions is true:

- The access unit includes a picture for each `nuh_layer_id` value in `TargetDecLayerIdList` and each picture in the access unit is an IDR picture.
- The access unit includes an IRAP picture with `nuh_layer_id` equal to `SmallestLayerId` for which `NoClrasOutputFlag` is equal to 1.

Any SPS NAL unit with any `nuh_layer_id` value containing the value of `sps_seq_parameter_set_id` for the active SPS RBSP for a particular layer shall have the same content as that of the active SPS RBSP for the particular layer unless it follows the access unit `auA` containing the last coded picture for which the active SPS RBSP for the particular layer is required to be active for the particular layer and precedes the first NAL unit succeeding `auA` in decoding order that activates an SPS RBSP with the same value of `seq_parameter_set_id`.

NOTE 2 – All SPSs, regardless of their values of `nuh_layer_id`, share the same value space for `sps_seq_parameter_set_id`. In other words, an SPS with `nuh_layer_id` equal to `X` and `sps_seq_parameter_set_id` equal to `A` would update the previously received SPS with `nuh_layer_id` not equal to `X` and `sps_seq_parameter_set_id` equal to `A`.

When a VPS RBSP (with a particular value of `vps_video_parameter_set_id`) is not already active and it is referred to by activation of an SPS RBSP (in which `sps_video_parameter_set_id` is equal to the `vps_video_parameter_set_id` value), or is referred to by an SEI NAL unit containing an active parameter sets SEI message (in which `active_video_parameter_set_id` is equal to the `vps_video_parameter_set_id` value), it is activated. This VPS RBSP is called the active VPS RBSP until it is deactivated by the activation of another VPS RBSP. A VPS RBSP, with that particular value of `vps_video_parameter_set_id`, shall be available to the decoding process prior to its activation, included in at least one access unit with `TemporalId` equal to 0 or provided through external means, and the VPS NAL unit containing the VPS RBSP shall have `nuh_layer_id` equal to 0 or `SmallestLayerId`.

An activated VPS RBSP shall remain active until the end of the bitstream or until it is deactivated by another VPS RBSP in an access unit in which at least one of the following conditions is true:

- The access unit includes a picture for each `nuh_layer_id` value in `TargetDecLayerIdList` and each picture in the access unit is an IDR picture.
- The access unit includes an IRAP picture with `nuh_layer_id` equal to `SmallestLayerId` for which `NoClrasOutputFlag` is equal to 1.

For any VPS NAL unit within the CVS with any value of `nuh_layer_id` and the value of `vps_video_parameter_set_id` equal to that of the active VPS RBSP for a CVS, the VPS RBSP in the VPS NAL unit shall have the same content as that of the active VPS RBSP for the CVS.

All constraints that are expressed on the relationship between the values of the syntax elements and the values of variables derived from those syntax elements in VPSs, SPSs and PPSs and other syntax elements are expressions of constraints that apply only to the active VPS RBSP, the active SPS RBSP for any particular layer and the active PPS RBSP for any particular layer. If any VPS RBSP, SPS RBSP and PPS RBSP is present that is never activated in the bitstream, its syntax elements shall have values that would conform to the specified constraints if it was activated by reference in an otherwise conforming bitstream.

During operation of the decoding process for NAL units of a non-base layer, the values of parameters of the active VPS RBSP, the active SPS RBSP for the non-base layer and the active PPS RBSP for the non-base layer are considered in effect. For interpretation of SEI messages applicable to a coded picture of a non-base layer, the values of the active VPS RBSP, the active SPS RBSP for the non-base layer and the active PPS RBSP for the non-base layer for the operation of the decoding process for the VCL NAL units of the coded picture are considered in effect unless otherwise specified in the SEI message semantics.

F.7.4.2.4.3 Order of access units association to CVSs

A bitstream conforming to this Specification consists of one or more CVSs.

A CVS consists of one or more access units. The order of NAL units and coded pictures and their association to access units is described in clause F.7.4.2.4.4.

The first access unit of a CVS is an IRAP access unit with NoRaslOutputFlag equal to 1.

It is a requirement of bitstream conformance that, when present, the next access unit after an access unit that contains an end of sequence NAL unit with nuh_layer_id equal to SmallestLayerId or an end of bitstream NAL unit shall be an IRAP access unit.

F.7.4.2.4.4 Order of NAL units and coded pictures and association to access units

This clause specifies the order of NAL units and coded pictures and their association to access unit for CVSs that contain NAL units with nuh_layer_id greater than 0 that are decoded using the decoding processes specified in Annexes F, G and H.

An access unit consists of one or more coded pictures, each with a distinct value of nuh_layer_id, and zero or more non-VCL NAL units. The association of VCL NAL units to coded pictures is described in clause 7.4.2.4.5.

The first access unit in the bitstream starts with the first NAL unit of the bitstream.

A VCL NAL unit is the first VCL NAL unit of an access unit, when all of the following conditions are true:

- first_slice_segment_in_pic_flag is equal to 1.
- At least one of the following conditions is true:
 - The previous picture in decoding order belongs to a different picture order count (POC) resetting period than the picture containing the VCL NAL unit.
 - PicOrderCntVal derived for the VCL NAL unit differs from the PicOrderCntVal of the previous picture in decoding order.

NOTE 1 – For any two consecutive access units auA and auB that belong to the same POC resetting period, the PicOrderCntVal of pictures in auA cannot be the same as the PicOrderCntVal of pictures in auB.

NOTE 2 – Additionally, more conditions, e.g., the following conditions, could but does not need to be used:

- The nuh_layer_id value of the VCL NAL unit is equal to SmallestLayerId.
- vps_poc_lsb_aligned_flag is equal to 1 and the slice_pic_order_cnt_lsb value of the VCL NAL unit differs from the slice_pic_order_cnt_lsb value of the previous VCL NAL unit in decoding order.

The first of any of the following NAL units preceding the first VCL NAL unit firstVclNalUnitInAu and succeeding the last VCL NAL unit preceding firstVclNalUnitInAu, if any, specifies the start of a new access unit:

- Access unit delimiter NAL unit (when present).
- VPS NAL unit (when present)
- SPS NAL unit (when present)
- PPS NAL unit (when present)
- Prefix SEI NAL unit (when present)
- NAL units with nal_unit_type in the range of RSV_NVCL41..RSV_NVCL44 (when present)
- NAL units with nal_unit_type in the range of UNSPEC48..UNSPEC55 (when present)

When there is none of the above NAL units preceding firstVclNalUnitInAu and succeeding the last VCL NAL unit preceding firstVclNalUnitInAu, if any, firstVclNalUnitInAu starts a new access unit.

A coded picture with nuh_layer_id equal to nuhLayerIdA shall precede, in decoding order, all coded pictures with nuh_layer_id greater than nuhLayerIdA in the same access unit.

The order of the coded pictures and non-VCL NAL units within an access unit shall obey the following constraints:

- When an access unit delimiter NAL unit is present, it shall be the first NAL unit. There shall be at most one access unit delimiter NAL unit in any access unit.
- When any VPS NAL units, SPS NAL units, PPS NAL units, prefix SEI NAL units, NAL units with nal_unit_type in the range of RSV_NVCL41..RSV_NVCL44, or NAL units with nal_unit_type in the range of UNSPEC48..UNSPEC55 are present, they shall not follow the last VCL NAL unit of the access unit.

- NAL units having `nal_unit_type` equal to `FD_NUT` or `SUFFIX_SEI_NUT`, or in the range of `RSV_NVCL45..RSV_NVCL47` or `UNSPEC56..UNSPEC63` shall not precede the first VCL NAL unit of the access unit.
- When an end of sequence NAL unit with `nuh_layer_id` `nuhLayerId` is present, it shall be the last NAL unit with `nuh_layer_id` equal to `nuhLayerId` in the access unit other than an end of bitstream NAL unit (when present).
- When an end of bitstream NAL unit is present, it shall be the last NAL unit in the access unit.

F.7.4.2.4.5 Order of VCL NAL units and association to coded pictures

The specifications in clause 7.4.2.4.5 apply.

F.7.4.3 Raw byte sequence payloads, trailing bits and byte alignment semantics

F.7.4.3.1 Video parameter set RBSP semantics

The specifications in clause 7.4.3.1 apply with the replacement of references to Annex C with references to clause F.13, with the replacement of the semantics of the syntax elements `vps_extension_flag` and `vps_extension_data_flag` with that specified below and with the following additions:

Each output operation point is associated with an operation point, a list of `nuh_layer_id` values of the output layers, in increasing order of `nuh_layer_id` values, denoted as `OpLayerIdList`, and the `OpTid` of the associated operation point. The `OpLayerIdList` of the operation point associated with an output operation point is also referred to as the `OpLayerIdList` of the output operation point.

`vps_extension_flag` equal to 0 specifies that no `vps_extension()` syntax structure is present in the VPS RBSP syntax structure. `vps_extension_flag` equal to 1 specifies that the `vps_extension()` syntax structure is present in the VPS RBSP syntax structure. When `MaxLayersMinus1` is greater than 0, `vps_extension_flag` shall be equal to 1.

`vps_extension_alignment_bit_equal_to_one` shall be equal to 1.

`vps_extension2_flag` equal to 0 specifies that no `vps_extension_data_flag` syntax elements are present in the VPS RBSP syntax structure. `vps_extension2_flag` equal to 1 specifies that `vps_extension_data_flag` syntax elements are present in the VPS RBSP syntax structure. Decoders conforming to a profile specified in Annexes A, G or H shall ignore all data that follow the value 1 for `vps_extension2_flag` in a VPS RBSP.

`vps_extension_data_flag` may have any value. Its presence and value do not affect the decoding process for the profiles specified in Annexes A, G or H. Decoders conforming to a profile specified in Annexes A, G or H shall ignore all `vps_extension_data_flag` syntax elements.

F.7.4.3.1.1 Video parameter set extension semantics

`splitting_flag` equal to 1 indicates that the `dimension_id[i][j]` syntax elements are not present and that the binary representation of the `nuh_layer_id` value in the NAL unit header are split into `NumScalabilityTypes` segments with lengths, in bits, according to the values of `dimension_id_len_minus1[j]` and that the values of `dimension_id[LayerIdxInVps[nuh_layer_id]][j]` are inferred from the `NumScalabilityTypes` segments. `splitting_flag` equal to 0 indicates that the syntax elements `dimension_id[i][j]` are present.

NOTE 1 – When `splitting_flag` is equal to 1, scalability identifiers of the present scalability dimensions can be derived from the `nuh_layer_id` syntax element in the NAL unit header by a bit masked copy. The respective bit mask for the *i*-th present scalability dimension is defined by the value of the `dimension_id_len_minus1[i]` syntax element and `dimBitOffset[i]` as specified in the semantics of `dimension_id_len_minus1[j]`.

`scalability_mask_flag[i]` equal to 1 indicates that `dimension_id` syntax elements corresponding to the *i*-th scalability dimension in Table F.1 are present. `scalability_mask_flag[i]` equal to 0 indicates that `dimension_id` syntax elements corresponding to the *i*-th scalability dimension are not present.

Table F.1 – Mapping of ScalabilityId to scalability dimensions

Scalability mask index	Scalability dimension	ScalabilityId mapping
0	Texture or depth	DepthLayerFlag
1	Multiview	ViewOrderIdx
2	Spatial/quality scalability	DependencyId
3	Auxiliary	AuxId
4-15	Reserved	

dimension_id_len_minus1[j] plus 1 specifies the length, in bits, of the **dimension_id**[i][j] syntax element.

When **splitting_flag** is equal to 1, the following applies:

- The variable **dimBitOffset**[0] is set equal to 0 and for j in the range of 1 to **NumScalabilityTypes** – 1, inclusive, **dimBitOffset**[j] is derived as follows:

$$\text{dimBitOffset}[j] = \sum_{\text{dimIdx}=0}^{j-1} (\text{dimension_id_len_minus1}[\text{dimIdx}] + 1) \quad (\text{F-2})$$

- The value of **dimension_id_len_minus1**[**NumScalabilityTypes** – 1] is inferred to be equal to 5 – **dimBitOffset**[**NumScalabilityTypes** – 1].
- The value of **dimBitOffset**[**NumScalabilityTypes**] is set equal to 6.

It is a requirement of bitstream conformance that when **NumScalabilityTypes** is greater than 0, **dimBitOffset**[**NumScalabilityTypes** – 1] shall be less than 6.

vps_nuh_layer_id_present_flag equal to 1 specifies that **layer_id_in_nuh**[i] for i from 1 to **MaxLayersMinus1**, inclusive, are present. **vps_nuh_layer_id_present_flag** equal to 0 specifies that **layer_id_in_nuh**[i] for i from 1 to **MaxLayersMinus1**, inclusive, are not present.

layer_id_in_nuh[i] specifies the value of the **nuh_layer_id** syntax element in VCL NAL units of the i-th layer. When i is greater than 0, **layer_id_in_nuh**[i] shall be greater than **layer_id_in_nuh**[i – 1]. For any value of i in the range of 0 to **MaxLayersMinus1**, inclusive, when not present, the value of **layer_id_in_nuh**[i] is inferred to be equal to i.

For i from 0 to **MaxLayersMinus1**, inclusive, the variable **LayerIdxInVps**[**layer_id_in_nuh**[i]] is set equal to i.

dimension_id[i][j] specifies the identifier of the j-th present scalability dimension type of the i-th layer. The number of bits used for the representation of **dimension_id**[i][j] is **dimension_id_len_minus1**[j] + 1 bits.

Depending on **splitting_flag**, the following applies:

- If **splitting_flag** is equal to 1, for i from 0 to **MaxLayersMinus1**, inclusive, and j from 0 to **NumScalabilityTypes** – 1, inclusive, **dimension_id**[i][j] is inferred to be equal to ((**layer_id_in_nuh**[i] & ((1 << **dimBitOffset**[j + 1]) – 1)) >> **dimBitOffset**[j]).
- Otherwise (**splitting_flag** is equal to 0), for j from 0 to **NumScalabilityTypes** – 1, inclusive, **dimension_id**[0][j] is inferred to be equal to 0.

The variable **ScalabilityId**[i][**smIdx**] specifying the identifier of the **smIdx**-th scalability dimension type of the i-th layer, and the variables **DepthLayerFlag**[**lId**], **ViewOrderIdx**[**lId**], **DependencyId**[**lId**], and **AuxId**[**lId**] specifying the depth flag, the view order index, the spatial/quality scalability identifier and the auxiliary identifier, respectively, of the layer with **nuh_layer_id** equal to **lId** are derived as follows:

```

NumViews = 1
for( i = 0; i <= MaxLayersMinus1; i++ ) {
    lId = layer_id_in_nuh[ i ]
    for( smIdx = 0; j = 0; smIdx < 16; smIdx++ ) {
        if( scalability_mask_flag[ smIdx ] )
            ScalabilityId[ i ][ smIdx ] = dimension_id[ i ][ j++ ]
        else
            ScalabilityId[ i ][ smIdx ] = 0
    }
    DepthLayerFlag[ lId ] = ScalabilityId[ i ][ 0 ]
    ViewOrderIdx[ lId ] = ScalabilityId[ i ][ 1 ]
    DependencyId[ lId ] = ScalabilityId[ i ][ 2 ]
    (F-3)
    AuxId[ lId ] = ScalabilityId[ i ][ 3 ]
    if( i > 0 ) {
        newViewFlag = 1
        for( j = 0; j < i; j++ )
            if( ViewOrderIdx[ lId ] == ViewOrderIdx[ layer_id_in_nuh[ j ] ] )
                newViewFlag = 0
        NumViews += newViewFlag
    }
}

```


AuxId[lld] equal to 0 specifies the layer with nuh_layer_id equal to lld does not contain auxiliary pictures. AuxId[lld] greater than 0 specifies the type of auxiliary pictures in layer with nuh_layer_id equal to lld as specified in Table F.2.

Table F.2 – Mapping of AuxId to the type of auxiliary pictures

AuxId	Name of AuxId	Type of auxiliary pictures	SEI message describing interpretation of auxiliary pictures
1	AUX_ALPHA	Alpha plane	Alpha channel information
2	AUX_DEPTH	Depth picture	Depth representation information
3..127		Reserved	
128..159		Unspecified	
160..255		Reserved	

NOTE 2 – The interpretation of auxiliary pictures associated with AuxId in the range of 128 to 159, inclusive, is specified through means other than the AuxId value.

AuxId[lld] shall be in the range of 0 to 2, inclusive, or 128 to 159, inclusive, for bitstreams conforming to this version of this Specification. Although the value of AuxId[lld] shall be in the range of 0 to 2, inclusive, or 128 to 159, inclusive, in this version of this Specification, decoders shall allow values of AuxId[lld] in the range of 0 to 255, inclusive.

It is a requirement of bitstream conformance that when AuxId[lld] is equal to AUX_ALPHA or AUX_DEPTH, either of the following applies:

- chroma_format_idc is equal to 0 in the active SPS for the layer with nuh_layer_id equal to lld.
- The value of all decoded chroma samples is equal to 1 << (BitDepth_C – 1) in all pictures that have nuh_layer_id equal to lld and for which this VPS RBSP is the active VPS RBSP.

SEI messages may describe the interpretation of auxiliary pictures, including their possible association with one or more primary pictures.

NOTE 3 – Unless constrained by the semantics of the SEI messages specifying the interpretation of auxiliary pictures, it is allowed to have two layers with nuh_layer_id values layerIdA and layerIdB such that AuxId[layerIdA] is equal to AuxId[layerIdB], both being greater than 0 and to have all values of ScalabilityId[LayerIdxInVps[layerIdA]][i] equal to ScalabilityId[LayerIdxInVps[layerIdB]][i] for each value of i in the range of 0 to 15, inclusive. SEI messages specifying the interpretation of auxiliary pictures may specify that a picture with nuh_layer_id equal to layerIdA and a picture with nuh_layer_id equal to layerIdB in the same access unit may both be associated with the same primary picture.

view_id_len specifies the length, in bits, of the view_id_val[i] syntax element.

view_id_val[i] specifies the view identifier of the i-th view specified by the VPS. The length of the view_id_val[i] syntax element is view_id_len bits. When not present, the value of view_id_val[i] is inferred to be equal to 0.

For each layer with nuh_layer_id equal to nuhLayerId, the value ViewId[nuhLayerId] is set equal to view_id_val[ViewOrderIdx[nuhLayerId]].

direct_dependency_flag[i][j] equal to 0 specifies that the layer with index j is not a direct reference layer for the layer with index i. **direct_dependency_flag[i][j]** equal to 1 specifies that the layer with index j is a direct reference layer for the layer with index i. When **direct_dependency_flag[i][j]** is not present for i and j in the range of 0 to MaxLayersMinus1, it is inferred to be equal to 0.

The variable DependencyFlag[i][j] is derived as follows:

```

for( i = 0; i <= MaxLayersMinus1; i++ )
    for( j = 0; j <= MaxLayersMinus1; j++ ) {
        DependencyFlag[ i ][ j ] = direct_dependency_flag[ i ][ j ]
        for( k = 0; k < i; k++ )
            if( direct_dependency_flag[ i ][ k ] && DependencyFlag[ k ][ j ] )
                DependencyFlag[ i ][ j ] = 1
    }

```

(F-4)

The variables NumDirectRefLayers[iNuhLid], IdDirectRefLayer[iNuhLid][d], NumRefLayers[iNuhLid], IdRefLayer[iNuhLid][r], NumPredictedLayers[iNuhLid] and IdPredictedLayer[iNuhLid][p] are derived as follows:

```

for( i = 0; i <= MaxLayersMinus1; i++ ) {
    iNuhLid = layer_id_in_nuh[ i ]
    for( j = 0, d = 0, r = 0, p = 0; j <= MaxLayersMinus1; j++ ) {
        jNuhLid = layer_id_in_nuh[ j ]

```

```

        if( direct_dependency_flag[ i ][ j ] )
            IdDirectRefLayer[ iNuhLid ][ d++ ] = jNuhLid
        if( DependencyFlag[ i ][ j ] )
            IdRefLayer[ iNuhLid ][ r++ ] = jNuhLid
        if( DependencyFlag[ j ][ i ] )
            IdPredictedLayer[ iNuhLid ][ p++ ] = jNuhLid
    }
    NumDirectRefLayers[ iNuhLid ] = d
    NumRefLayers[ iNuhLid ] = r
    NumPredictedLayers[ iNuhLid ] = p
}

```

The variables NumIndependentLayers, NumLayersInTreePartition[i] and TreePartitionLayerIdList[i][j] for i in the range of 0 to NumIndependentLayers – 1, inclusive, and j in the range of 0 to NumLayersInTreePartition[i] – 1, inclusive, are derived as follows:

```

for( i = 0; i <= 63; i++ )
    layerIdInListFlag[ i ] = 0
for( i = 0, k = 0; i <= MaxLayersMinus1; i++ ) {
    iNuhLid = layer_id_in_nuh[ i ]
    if( NumDirectRefLayers[ iNuhLid ] == 0 ) {
        TreePartitionLayerIdList[ k ][ 0 ] = iNuhLid
        for( j = 0, h = 1; j < NumPredictedLayers[ iNuhLid ]; j++ ) {
            predLid = IdPredictedLayer[ iNuhLid ][ j ]
            if( !layerIdInListFlag[ predLid ] ) {
                TreePartitionLayerIdList[ k ][ h++ ] = predLid
                layerIdInListFlag[ predLid ] = 1
            }
        }
        NumLayersInTreePartition[ k++ ] = h
    }
}
NumIndependentLayers = k

```

It is a requirement of bitstream conformance that AuxId[IdDirectRefLayer[nuhLayerIdA][j]] for any values of nuhLayerIdA and j shall be equal to AuxId[nuhLayerIdA], when AuxId[nuhLayerIdA] is in the range of 0 to 2, inclusive.

NOTE 4 – In other words, no prediction takes place between layers with a different value of AuxId, when AuxId is in the range of 0 to 2, inclusive.

num_add_layer_sets specifies the number of additional layer sets. The value of num_add_layer_sets shall be in the range of 0 to 1 023, inclusive. When vps_base_layer_available_flag is equal to 0, the value of num_add_layer_sets shall be greater than 0. When not present, the value of num_add_layer_sets is inferred to be equal to 0.

The variable NumLayerSets is derived as follows:

$$\text{NumLayerSets} = \text{vps_num_layer_sets_minus1} + 1 + \text{num_add_layer_sets} \quad (\text{F-7})$$

When num_add_layer_sets is greater than 0, the variables FirstAddLayerSetIdx and LastAddLayerSetIdx are derived as follows:

$$\begin{aligned} \text{FirstAddLayerSetIdx} &= \text{vps_num_layer_sets_minus1} + 1 \\ \text{LastAddLayerSetIdx} &= \text{FirstAddLayerSetIdx} + \text{num_add_layer_sets} - 1 \end{aligned} \quad (\text{F-8})$$

highest_layer_idx_plus1[i][j] specifies the values of NumLayersInIdList[vps_num_layer_sets_minus1 + 1 + i] and LayerSetLayerIdList[vps_num_layer_sets_minus1 + 1 + i][layerNum] as follows:

```

layerNum = 0
lsIdx = vps_num_layer_sets_minus1 + 1 + i
for( treeIdx = 1; treeIdx < NumIndependentLayers; treeIdx++ )
    for( layerCnt = 0; layerCnt < highest_layer_idx_plus1[ i ][ treeIdx ]; layerCnt++ )
        LayerSetLayerIdList[ lsIdx ][ layerNum++ ] =
            TreePartitionLayerIdList[ treeIdx ][ layerCnt ]
    NumLayersInIdList[ lsIdx ] = layerNum

```

The value of `highest_layer_idx_plus1[i][j]` shall be in the range of 0 to `NumLayersInTreePartition[j]`, inclusive.

The length of the `highest_layer_idx_plus1[i][j]` syntax element is $\text{Ceil}(\log_2(\text{NumLayersInTreePartition}[j] + 1))$ bits.

It is a requirement of bitstream conformance that `NumLayersInIdList[vps_num_layer_sets_minus1 + 1 + i]` shall be greater than 0.

vps_sub_layers_max_minus1_present_flag equal to 1 specifies that the syntax elements `sub_layers_vps_max_minus1[i]` are present. **vps_sub_layers_max_minus1_present_flag** equal to 0 specifies that the syntax elements `sub_layers_vps_max_minus1[i]` are not present.

sub_layers_vps_max_minus1[i] plus 1 specifies, for each value of *i* in the range of (`vps_base_layer_internal_flag ? 0 : 1`) to `MaxLayersMinus1`, inclusive, the maximum number of temporal sub-layers that may be present in the CVS for the layer with `nuh_layer_id` `layerId` equal to `layer_id_in_nuh[i]`. When `vps_base_layer_internal_flag` is equal to 0, `sub_layers_vps_max_minus1[0]` constrains the access units for which a decoded picture with `nuh_layer_id` equal to 0 may be provided by external means as follows: a decoded picture with `nuh_layer_id` equal to 0 cannot be provided by external means for decoding of an access unit with `TemporalId` greater than `sub_layers_vps_max_minus1[0]`. The value of `sub_layers_vps_max_minus1[i]` shall be in the range of 0 to `vps_max_sub_layers_minus1`, inclusive. When not present, the value of `sub_layers_vps_max_minus1[i]` is inferred to be equal to `vps_max_sub_layers_minus1`.

The variable `MaxSubLayersInLayerSetMinus1[i]` is derived as follows:

```
for( i = 0; i < NumLayerSets; i++ ) {
    maxSMinus1 = 0
    for( k = 0; k < NumLayersInIdList[i]; k++ ) {
        lId = LayerSetLayerIdList[i][k]
        maxSMinus1 = Max( maxSMinus1,
            sub_layers_vps_max_minus1[LayerIdxInVps[lId]] )
    }
    MaxSubLayersInLayerSetMinus1[i] = maxSMinus1
}
```

(F-10)

max_tid_ref_present_flag equal to 1 specifies that the syntax element `max_tid_il_ref_pics_plus1[i][j]` is present. **max_tid_ref_present_flag** equal to 0 specifies that the syntax element `max_tid_il_ref_pics_plus1[i][j]` is not present.

max_tid_il_ref_pics_plus1[i][j] equal to 0 specifies that non-IRAP pictures with `nuh_layer_id` equal to `layer_id_in_nuh[i]` are not used as source pictures for inter-layer prediction for pictures with `nuh_layer_id` equal to `layer_id_in_nuh[j]`. **max_tid_il_ref_pics_plus1[i][j]** greater than 0 specifies that pictures with `nuh_layer_id` equal to `layer_id_in_nuh[i]` and `TemporalId` greater than `max_tid_il_ref_pics_plus1[i][j] - 1` are not used as source pictures for inter-layer prediction for pictures with `nuh_layer_id` equal to `layer_id_in_nuh[j]`. When not present, the value of `max_tid_il_ref_pics_plus1[i][j]` is inferred to be equal to 7.

default_ref_layers_active_flag equal to 1 specifies that for each picture referring to the VPS, the direct reference layer pictures that belong to all direct reference layers of the layer containing the picture and that may be used for inter-layer prediction as specified by the values of `sub_layers_vps_max_minus1[i]` and `max_tid_il_ref_pics_plus1[i][j]` are present in the same access unit as the picture and are included in the inter-layer reference picture set of the picture. **default_ref_layers_active_flag** equal to 0 specifies that the above restriction may or may not apply.

vps_num_profile_tier_level_minus1 plus 1 specifies the number of `profile_tier_level()` syntax structures in the VPS. The value of `vps_num_profile_tier_level_minus1` shall be in the range of 0 to 63, inclusive. When `vps_max_layers_minus1` is greater than 0, the value of `vps_num_profile_tier_level_minus1` shall be greater than or equal to 1.

vps_profile_present_flag[i] equal to 1 specifies that profile and tier information is present in the *i*-th `profile_tier_level()` syntax structure. **vps_profile_present_flag[i]** equal to 0 specifies that profile and tier information is not present in the *i*-th `profile_tier_level()` syntax structure and is inferred as specified in clause F.7.4.4.

num_add_olss specifies the number of OLSs in addition to the first `NumLayerSets` OLSs specified by the VPS. The value of `num_add_olss` shall be in the range of 0 to 1 023, inclusive. When not present, the value of `num_add_olss` is inferred to be equal to 0.

default_output_layer_idc specifies the derivation of the output layers for the OLSs with index in the range of 1 to `vps_num_layer_sets_minus1`, inclusive. **default_output_layer_idc** equal to 0 specifies that all layers in each of the OLSs with index in the range of 1 to `vps_num_layer_sets_minus1`, inclusive, are output layers of their respective OLSs. **default_output_layer_idc** equal to 1 specifies that only the layer with the highest value of `nuh_layer_id` such that `nuh_layer_id` equal to `nuhLayerIdA` in each of the OLSs with index in the range of 1 to `vps_num_layer_sets_minus1`, inclusive, is an output layer of its OLS. **default_output_layer_idc** equal to 2 specifies that the output layers for the OLSs with index in the range of 1 to `vps_num_layer_sets_minus1`, inclusive, are specified with the syntax elements

output_layer_flag[i][j]. The value of 3 for default_output_layer_idc is reserved for future use by ITU-T | ISO/IEC. Although the value of default_output_layer_idc is required to be less than 3 in this version of this Specification, decoders shall allow a value of default_output_layer_idc equal to 3 to appear in the syntax.

The variable defaultOutputLayerIdc is set equal to Min(default_output_layer_idc, 2).

layer_set_idx_for_ols_minus1[i] plus 1 specifies the index of the layer set for the i-th OLS. The value of layer_set_idx_for_ols_minus1[i] shall be in the range of 0 to NumLayerSets – 2, inclusive. The length of the layer_set_idx_for_ols_minus1[i] syntax element is Ceil(Log2(NumLayerSets – 1)) bits. When not present, the value of layer_set_idx_for_ols_minus1[i] is inferred to be equal to 0.

For i in the range of 0 to NumOutputLayerSets – 1, inclusive, the variable OlsIdxToLsIdx[i] is derived as specified in the following:

$$\text{OlsIdxToLsIdx}[i] = (i < \text{NumLayerSets}) ? i : (\text{layer_set_idx_for_ols_minus1}[i] + 1) \quad (\text{F-11})$$

output_layer_flag[i][j] equal to 1 specifies that the j-th layer in the i-th OLS is an output layer. output_layer_flag[i][j] equal to 0 specifies that the j-th layer in the i-th OLS is not an output layer.

The value of output_layer_flag[0][0] is inferred to be equal to 1.

When defaultOutputLayerIdc is equal to 0 or 1, for i in the range of 0 to vps_num_layer_sets_minus1, inclusive, and j in the range of 0 to NumLayersInIdList[OlsIdxToLsIdx[i]] – 1, inclusive, the variable OutputLayerFlag[i][j] is derived as follows:

- If defaultOutputLayerIdc is equal to 0 or LayerSetLayerIdList[OlsIdxToLsIdx[i]][j] is equal to nuhLayerIdA, with nuhLayerIdA being the highest value in LayerSetLayerIdList[OlsIdxToLsIdx[i]], OutputLayerFlag[i][j] is set equal to 1.
- Otherwise, OutputLayerFlag[i][j] is set equal to 0.

For i in the range of (defaultOutputLayerIdc == 2) ? 0 : (vps_num_layer_sets_minus1 + 1) to NumOutputLayerSets – 1, inclusive, and j in the range of 0 to NumLayersInIdList[OlsIdxToLsIdx[i]] – 1, inclusive, the variable OutputLayerFlag[i][j] is set equal to output_layer_flag[i][j].

The variable NumOutputLayersInOutputLayerSet[i] is derived as follows:

$$\begin{aligned} \text{NumOutputLayersInOutputLayerSet}[i] &= 0 \\ \text{for}(j = 0; j < \text{NumLayersInIdList}[\text{OlsIdxToLsIdx}[i]]; j++) \{ \\ \quad \text{NumOutputLayersInOutputLayerSet}[i] &+= \text{OutputLayerFlag}[i][j] \\ \quad \text{if}(\text{OutputLayerFlag}[i][j]) \\ \quad \quad \text{OlsHighestOutputLayerId}[i] &= \text{LayerSetLayerIdList}[\text{OlsIdxToLsIdx}[i]][j] \end{aligned} \quad (\text{F-12})$$

It is a requirement of bitstream conformance that NumOutputLayersInOutputLayerSet[i] shall be greater than 0 for i in the range of 0 to NumOutputLayerSets – 1, inclusive.

The variables NumNecessaryLayers[olsIdx] and NecessaryLayerFlag[olsIdx][lIdx] are derived as follows:

$$\begin{aligned} \text{for}(\text{olsIdx} = 0; \text{olsIdx} < \text{NumOutputLayerSets}; \text{olsIdx}++) \{ \\ \quad \text{lsIdx} &= \text{OlsIdxToLsIdx}[\text{olsIdx}] \\ \quad \text{for}(\text{lsLayerIdx} = 0; \text{lsLayerIdx} < \text{NumLayersInIdList}[\text{lsIdx}]; \text{lsLayerIdx}++) \\ \quad \quad \text{NecessaryLayerFlag}[\text{olsIdx}][\text{lsLayerIdx}] &= 0 \\ \quad \text{for}(\text{lsLayerIdx} = 0; \text{lsLayerIdx} < \text{NumLayersInIdList}[\text{lsIdx}]; \text{lsLayerIdx}++) \\ \quad \quad \text{if}(\text{OutputLayerFlag}[\text{olsIdx}][\text{lsLayerIdx}]) \{ \\ \quad \quad \quad \text{NecessaryLayerFlag}[\text{olsIdx}][\text{lsLayerIdx}] &= 1 \\ \quad \quad \quad \text{currLayerId} &= \text{LayerSetLayerIdList}[\text{lsIdx}][\text{lsLayerIdx}] \\ \quad \quad \quad \text{for}(\text{rLsLayerIdx} = 0; \text{rLsLayerIdx} < \text{lsLayerIdx}; \text{rLsLayerIdx}++) \{ \\ \quad \quad \quad \quad \text{refLayerId} &= \text{LayerSetLayerIdList}[\text{lsIdx}][\text{rLsLayerIdx}] \\ \\ \quad \quad \quad \text{if}(\text{DependencyFlag}[\text{LayerIdxInVps}[\text{currLayerId}]][\text{LayerIdxInVps}[\text{refLayerId}]]) \\ \quad \quad \quad \quad \text{NecessaryLayerFlag}[\text{olsIdx}][\text{rLsLayerIdx}] &= 1 \\ \quad \quad \quad \} \\ \quad \quad \} \\ \quad \text{NumNecessaryLayers}[\text{olsIdx}] &= 0 \\ \quad \text{for}(\text{lsLayerIdx} = 0; \text{lsLayerIdx} < \text{NumLayersInIdList}[\text{lsIdx}]; \text{lsLayerIdx}++) \\ \quad \quad \text{NumNecessaryLayers}[\text{olsIdx}] &+= \text{NecessaryLayerFlag}[\text{olsIdx}][\text{lsLayerIdx}] \\ \} \end{aligned} \quad (\text{F-13})$$

It is a requirement of bitstream conformance that for each layer index `layerIdx` in the range of (`vps_base_layer_internal_flag ? 0 : 1`) to `MaxLayersMinus1`, inclusive, there shall be at least one OLS with index `olsIdx` such that `NecessaryLayerFlag[olsIdx][lsLayerIdx]` is equal to 1 for the value of `lsLayerIdx` for which `LayerSetLayerIdList[OlsIdxToLsIdx[olsIdx]][lsLayerIdx]` is equal to `layer_id_in_nuh[layerIdx]`.

NOTE 5 – In other words, each layer has to be an output layer or a reference layer of an output layer in at least one OLS.

It is a requirement of bitstream conformance that when `num_add_layer_sets` is greater than 0 and `olsIdx` has any such value that `OlsIdxToLsIdx[olsIdx]` is in the range of `FirstAddLayerSetIdx` to `LastAddLayerSetIdx`, inclusive, and `NumNecessaryLayers[olsIdx]` is equal to 1, `NecessaryLayerFlag[olsIdx][0]` shall be equal to 1.

NOTE 6 – In other words, when an additional layer set has exactly one necessary layer, the necessary layer is required to be the layer with the smallest `nuh_layer_id` value within the additional layer set.

profile_tier_level_idx[i][j] specifies the index, into the list of `profile_tier_level()` syntax structures in the VPS, of the `profile_tier_level()` syntax structure that applies to the `j`-th layer of the `i`-th OLS as specified in clause F.7.4.4. The length of the `profile_tier_level_idx[i][j]` syntax element is `Ceil(Log2(vps_num_profile_tier_level_minus1 + 1))` bits.

When `NecessaryLayerFlag[i][j]` is equal to 1 and `vps_num_profile_tier_level_minus1` is equal to 0, the value of `profile_tier_level_idx[i][j]` is inferred to be equal to 0.

When `vps_base_layer_internal_flag` is equal to 1, the following applies:

- If `vps_max_layers_minus1` is greater than 0, the value of `profile_tier_level_idx[0][0]` is inferred to be equal to 1.
- Otherwise (`vps_max_layers_minus1` is equal to 0), the value of `profile_tier_level_idx[0][0]` is inferred to be equal to 0.

When present, the value of `profile_tier_level_idx[i][j]` shall be in the range of (`vps_base_layer_internal_flag ? 0 : 1`) to `vps_num_profile_tier_level_minus1`, inclusive.

alt_output_layer_flag[i] equal to 0 specifies that an alternative output layer is not used for any output layer in the `i`-th OLS. **alt_output_layer_flag[i]** equal to 1 specifies that an alternative output layer may be used for the output layer in the `i`-th OLS.

When not present, the value of `alt_output_layer_flag[i]` is inferred to be equal to 0.

NOTE 7 – When `alt_output_layer_flag[olsIdx]` is equal to 0, pictures that do not belong to the output layers of the OLS with index `olsIdx` are not output. When `alt_output_layer_flag[olsIdx]` is equal to 1 and a picture belonging to the output layer of the OLS with index `olsIdx` is not present in an access unit or has `PicOutputFlag` equal to 0, a picture with the highest `nuh_layer_id` among those pictures of the access unit for which `PicOutputFlag` is equal to 1 and having the `nuh_layer_id` value among the `nuh_layer_id` values of the reference layers of the output layer is output.

For each value of `olsIdx` in the range of 0 to `NumOutputLayerSets – 1`, inclusive, the following applies:

- When `alt_output_layer_flag[olsIdx]` is equal to 1, the value of `pic_output_flag` shall be the same in the slice headers of an access unit that have `nuh_layer_id` value equal to `OlsHighestOutputLayerId[olsIdx]` or equal to the `nuh_layer_id` value of any reference layer of the layer with `nuh_layer_id` equal to `OlsHighestOutputLayerId[olsIdx]`.
- The variable `olsBitstream` is derived as follows:
 - Let `lsIdx` be equal to `OlsIdxToLsIdx[olsIdx]`.
 - If `lsIdx` is less than or equal to `vps_num_layer_sets_minus1`, let `olsBitstream` be the output of the sub-bitstream extraction process specified in clause F.10.1 with the following inputs: the current bitstream, `tIdTarget` equal to 6 and `layerIdListTarget` equal to `LayerSetLayerIdList[lsIdx]`.
 - Otherwise, let `olsBitstream` be the output of the sub-bitstream extraction process specified in clause F.10.3 with the following inputs: the current bitstream, `tIdTarget` equal to 6 and `layerIdListTarget` equal to `LayerSetLayerIdList[lsIdx]`.
- Let `truncatedOlsBitstream` be `olsBitstream` or be formed from the `olsBitstream` by removing access units preceding, in decoding order, any access unit with an IRAP picture having `nuh_layer_id` equal to `SmallestLayerId`.
- It is a requirement of bitstream conformance that when `alt_output_layer_flag[olsIdx]` is equal to 1, a bitstream that is formed by removing, from the `truncatedOlsBitstream`, any coded picture that is not used as a reference for prediction for any other picture and is not the only coded picture of an access unit is a conforming bitstream.

NOTE 8 – When `alt_output_layer_flag[olsIdx]` is equal to 1, encoders are required to set the values of `max_vps_dec_pic_buffering_minus1[i][k][j]` such that these values suffice also when pictures of an alternative output layer are marked as "needed for output" in the HRD.

vps_num_rep_formats_minus1 plus 1 specifies the number of the following `rep_format()` syntax structures in the VPS. The value of `vps_num_rep_formats_minus1` shall be in the range of 0 to 255, inclusive.

rep_format_idx_present_flag equal to 1 specifies that the syntax elements `vps_rep_format_idx[i]` are present. `rep_format_idx_present_flag` equal to 0 specifies that the syntax elements `vps_rep_format_idx[i]` are not present. When not present, the value of `rep_format_idx_present_flag` is inferred to be equal to 0.

vps_rep_format_idx[i] specifies the index, into the list of `rep_format()` syntax structures in the VPS, of the `rep_format()` syntax structure that applies to the layer with `nuh_layer_id` equal to `layer_id_in_nuh[i]`. When not present, the value of `vps_rep_format_idx[i]` is inferred to be equal to $\text{Min}(i, \text{vps_num_rep_formats_minus1})$. The value of `vps_rep_format_idx[i]` shall be in the range of 0 to `vps_num_rep_formats_minus1`, inclusive. The number of bits used for the representation of `vps_rep_format_idx[i]` is $\text{Ceil}(\text{Log2}(\text{vps_num_rep_formats_minus1} + 1))$.

max_one_active_ref_layer_flag equal to 1 specifies that at most one picture is used for inter-layer prediction for each picture in the CVS. `max_one_active_ref_layer_flag` equal to 0 specifies that more than one picture may be used for inter-layer prediction for each picture in the CVS.

vps_poc_lsb_aligned_flag equal to 0 specifies that the value of `slice_pic_order_cnt_lsb` may or may not be the same in different pictures of an access unit. `vps_poc_lsb_aligned_flag` equal to 1 specifies that the value of `slice_pic_order_cnt_lsb` is the same in all pictures of an access unit. Additionally, the value of `vps_poc_lsb_aligned_flag` affects the decoding process for picture order count in clause F.8.3.1. When not present, the value of `vps_poc_lsb_aligned_flag` is inferred to be equal to 0.

poc_lsb_not_present_flag[i] equal to 1 specifies that the `slice_pic_order_cnt_lsb` syntax element is not present in the slice headers of IDR pictures with `nuh_layer_id` equal to `layer_id_in_nuh[i]` in the CVS. `poc_lsb_not_present_flag[i]` equal to 0 specifies that `slice_pic_order_cnt_lsb` syntax element may or may not be present in the slice headers of IDR pictures with `nuh_layer_id` equal to `layer_id_in_nuh[i]` in the CVS. When not present, the value of `poc_lsb_not_present_flag[i]` is inferred to be equal to 0.

direct_dep_type_len_minus2 plus 2 specifies the number of bits of the `direct_dependency_type[i][j]` and the `direct_dependency_all_layers_type` syntax elements. In bitstreams conforming to this version of this Specification the value of `direct_dep_type_len_minus2` shall be equal to 0 or 1. Although the value of `direct_dep_type_len_minus2` shall be equal to 0 or 1 in this version of this Specification, decoders shall allow other values of `direct_dep_type_len_minus2` in the range of 0 to 30, inclusive, to appear in the syntax.

direct_dependency_all_layers_flag equal to 1 specifies that the syntax element `direct_dependency_type[i][j]` is not present and inferred from `direct_dependency_all_layers_type`. `direct_dependency_all_layers_flag` equal to 0 indicates that the syntax element `direct_dependency_type[i][j]` is present.

direct_dependency_all_layers_type, when present, specifies the inferred value of `direct_dependency_type[i][j]` for all combinations of *i*-th and *j*-th layers. The length of the `direct_dependency_all_layers_type` syntax element is `direct_dep_type_len_minus2` + 2 bits. Although the value of `direct_dependency_all_layers_type` is required to be in the range of 0 to 6, inclusive, in this version of this Specification, decoders shall allow values of `direct_dependency_all_layers_type` in the range of 0 to $2^{32} - 2$, inclusive, to appear in the syntax.

direct_dependency_type[i][j] indicates the type of dependency between the layer with `nuh_layer_id` equal to `layer_id_in_nuh[i]` and the layer with `nuh_layer_id` equal to `layer_id_in_nuh[j]`. `direct_dependency_type[i][j]` equal to 0 specifies that the layer with `nuh_layer_id` equal to `layer_id_in_nuh[j]` may be used for inter-layer sample prediction but is not used for inter-layer motion prediction of the layer with `nuh_layer_id` equal to `layer_id_in_nuh[i]`. `direct_dependency_type[i][j]` equal to 1 specifies that the layer with `nuh_layer_id` equal to `layer_id_in_nuh[j]` may be used for inter-layer motion prediction but is not used for inter-layer sample prediction of the layer with `nuh_layer_id` equal to `layer_id_in_nuh[i]`. `direct_dependency_type[i][j]` equal to 2 specifies that the layer with `nuh_layer_id` equal to `layer_id_in_nuh[j]` may be used for both inter-layer motion prediction and inter-layer sample prediction of the layer with `nuh_layer_id` equal to `layer_id_in_nuh[i]`. The length of the `direct_dependency_type[i][j]` syntax element is `direct_dep_type_len_minus2` + 2 bits. Although the value of `direct_dependency_type[i][j]` shall be in the range of 0 to 2, inclusive, when the layer with `nuh_layer_id` equal to `layer_id_in_nuh[i]` conforms to a profile specified in Annexes A, G or H, and in the range of 0 to 6, inclusive, when the layer with `nuh_layer_id` equal to `layer_id_in_nuh[i]` conforms to a profile specified in Annex I, decoders shall allow values of `direct_dependency_type[i][j]` in the range of 0 to $2^{32} - 2$, inclusive, to appear in the syntax.

When `direct_dependency_all_layers_flag` is equal to 1, for any *i* in the range of $(1 - \text{vps_base_layer_internal_flag}) + 1$ to `MaxLayersMinus1`, inclusive, and any *j* in the range of $1 - \text{vps_base_layer_internal_flag}$ to *i* - 1, inclusive, when `direct_dependency_flag[i][j]` is equal to 1, the value of `direct_dependency_type[i][j]` is inferred to be equal to `direct_dependency_all_layers_type`.

When `vps_base_layer_internal_flag` is equal to 0, for any *i* in the range of 1 to `MaxLayersMinus1`, inclusive, when `direct_dependency_flag[i][0]` is equal to 1, the value of `direct_dependency_type[i][0]` is inferred to be equal to 0.

The variables `VpsInterLayerSamplePredictionEnabled[i][j]` and `VpsInterLayerMotionPredictionEnabled[i][j]` are derived as follows:

```

if( direct_dependency_flag[ i ][ j ] )
    VpsInterLayerSamplePredictionEnabled[ i ][ j ] = ( direct_dependency_type[ i ][ j ] + 1 ) & 0x1
else
    VpsInterLayerSamplePredictionEnabled[ i ][ j ] = 0
if( direct_dependency_flag[ i ][ j ] )
    VpsInterLayerMotionPredictionEnabled[ i ][ j ] = ( ( direct_dependency_type[ i ][ j ] + 1 ) & 0x2 ) ?
1 : 0
else
    VpsInterLayerMotionPredictionEnabled[ i ][ j ] = 0

```

(F-14)

vps_non_vui_extension_length specifies the length of the non-VUI VPS extension data following this syntax element and before `vps_vui_present_flag`, in bytes. The value of `vps_non_vui_extension_length` shall be in the range of 0 to 4096, inclusive.

vps_non_vui_extension_data_byte may have any value. Decoders shall ignore the value of `vps_non_vui_extension_data_byte`. Its value does not affect the decoding process specified in this version of this Specification.

vps_vui_present_flag equal to 1 specifies that the `vps_vui()` syntax structure is present in the VPS. `vps_vui_present_flag` equal to 0 specifies that the `vps_vui()` syntax structure is not present in the VPS.

vps_vui_alignment_bit_equal_to_one shall be equal to 1.

F.7.4.3.1.2 Representation format semantics

chroma_and_bit_depth_vps_present_flag equal to 1 specifies that the syntax elements `chroma_format_vps_idc`, `bit_depth_vps_luma_minus8` and `bit_depth_vps_chroma_minus8` are present and that the syntax element `separate_colour_plane_vps_flag` may be present in the `rep_format()` structure. `chroma_and_bit_depth_vps_present_flag` equal to 0 specifies that the syntax elements `chroma_format_vps_idc`, `separate_colour_plane_vps_flag`, `bit_depth_vps_luma_minus8` and `bit_depth_vps_chroma_minus8` are not present and are inferred from the previous `rep_format()` syntax structure in the VPS. The value of `chroma_and_bit_depth_vps_present_flag` of the first `rep_format()` syntax structure in the VPS shall be equal to 1.

pic_width_vps_in_luma_samples, **pic_height_vps_in_luma_samples**, **chroma_format_vps_idc**, **separate_colour_plane_vps_flag**, **bit_depth_vps_luma_minus8** and **bit_depth_vps_chroma_minus8** are used for inference of the values of the SPS syntax elements `pic_width_in_luma_samples`, `pic_height_in_luma_samples`, `chroma_format_idc`, `separate_colour_plane_flag`, `bit_depth_luma_minus8` and `bit_depth_chroma_minus8`, respectively, for each SPS that refers to the VPS. When not present in the *i*-th `rep_format()` syntax structure in the VPS, the value of each of these syntax elements is inferred to be equal to the value of the corresponding syntax element in the (*i* – 1)-th `rep_format()` syntax structure in the VPS. `pic_width_vps_in_luma_samples` shall not be equal to 0 and shall be an integer multiple of `MinCbSizeY`. `pic_height_vps_in_luma_samples` shall not be equal to 0 and shall be an integer multiple of `MinCbSizeY`. The value of `chroma_format_vps_idc` shall be in the range of 0 to 3, inclusive. `bit_depth_vps_luma_minus8` shall be in the range of 0 to 8, inclusive. `bit_depth_vps_chroma_minus8` shall be in the range of 0 to 8, inclusive.

conformance_window_vps_flag equal to 1 specifies that the syntax elements `conf_win_vps_left_offset`, `conf_win_vps_right_offset`, `conf_win_vps_top_offset` and `conf_win_vps_bottom_offset` follow next in the `rep_format()` structure. `conformance_window_vps_flag` equal to 0 specifies that the syntax elements `conf_win_vps_left_offset`, `conf_win_vps_right_offset`, `conf_win_vps_top_offset` and `conf_win_vps_bottom_offset` are not present.

conf_win_vps_left_offset, **conf_win_vps_right_offset**, **conf_win_vps_top_offset** and **conf_win_vps_bottom_offset** are used for inference of the values of the SPS syntax elements `conf_win_left_offset`, `conf_win_right_offset`, `conf_win_top_offset` and `conf_win_bottom_offset`, respectively, for each SPS that refers to the VPS. When not present, the values of `conf_win_vps_left_offset`, `conf_win_vps_right_offset`, `conf_win_vps_top_offset` and `conf_win_vps_bottom_offset` are inferred to be equal to 0.

The value of `SubWidthC * (conf_win_vps_left_offset + conf_win_vps_right_offset)` shall be less than `pic_width_vps_in_luma_samples` and the value of `SubHeightC * (conf_win_vps_top_offset + conf_win_vps_bottom_offset)` shall be less than `pic_height_vps_in_luma_samples`.

F.7.4.3.1.3 DPB size semantics

For the *l*Idx-th layer set, the number of sub-DPBs is `NumLayersInIdList[lIdx]` and for each layer with a particular value of `nuh_layer_id` in the layer set, the sub-DPB with index `layerIdx` is assigned, where `LayerSetLayerIdList[lIdx][layerIdx]` is equal to `nuh_layer_id`.

sub_layer_flag_info_present_flag[i][j] equal to 1 specifies that **sub_layer_dpb_info_present_flag**[i][j] is present for j in the range of 1 to **MaxSubLayersInLayerSetMinus1**[**OlsIdxToLsIdx**[i]], inclusive. **sub_layer_flag_info_present_flag**[i] equal to 0 specifies that, for each value of j greater than 0, **sub_layer_dpb_info_present_flag**[i][j] is not present.

sub_layer_dpb_info_present_flag[i][j] equal to 1 specifies that **max_vps_dec_pic_buffering_minus1**[i][k][j] may be present for k in the range of 0 to **NumLayersInIdList**[**OlsIdxToLsIdx**[i]] - 1, inclusive, for the j-th sub-layer of the i-th OLS, and **max_vps_num_reorder_pics**[i][j] and **max_vps_latency_increase_plus1**[i][j] are present for the j-th sub-layer of the i-th OLS. **sub_layer_dpb_info_present_flag**[i][j] equal to 0 specifies that **max_vps_dec_pic_buffering_minus1**[i][k][j] for k in the range of 0 to **NumLayersInIdList**[**OlsIdxToLsIdx**[i]] - 1, inclusive, **max_vps_num_reorder_pics**[i][j] and **max_vps_latency_increase_plus1**[i][j] are not present. The value of **sub_layer_dpb_info_present_flag**[i][0] for any possible value of i is inferred to be equal to 1. When not present, the value of **sub_layer_dpb_info_present_flag**[i][j] for j greater than 0 and any possible value of i, is inferred to be equal to 0.

max_vps_dec_pic_buffering_minus1[i][k][j] plus 1, when **NecessaryLayerFlag**[i][k] is equal to 1, specifies the maximum number of decoded pictures, of the k-th layer in the i-th OLS for the CVS, that need to be stored in the DPB when **HighestTid** is equal to j.

When **NecessaryLayerFlag**[i][k] is equal to 1, the following applies for i in the range of 1 to **NumOutputLayerSets** - 1, inclusive:

- When j is greater than 0, **max_vps_dec_pic_buffering_minus1**[i][k][j] shall be greater than or equal to **max_vps_dec_pic_buffering_minus1**[i][k][j - 1].
- When **max_vps_dec_pic_buffering_minus1**[i][0][0] is not present, it is inferred to be equal to 0.
- When **max_vps_dec_pic_buffering_minus1**[i][k][j] is not present for j in the range of 1 to **MaxSubLayersInLayerSetMinus1**[**OlsIdxToLsIdx**[i]], inclusive, it is inferred to be equal to **max_vps_dec_pic_buffering_minus1**[i][k][j - 1].

max_vps_num_reorder_pics[i][j] specifies, when **HighestTid** is equal to j, the maximum allowed number of access units containing a picture with **PicOutputFlag** equal to 1 that can precede any access unit auA that contains a picture with **PicOutputFlag** equal to 1 in the i-th OLS in the CVS in decoding order and follow the access unit auA that contains a picture with **PicOutputFlag** equal to 1 in output order. When **max_vps_num_reorder_pics**[i][j] is not present for j in the range of 1 to **MaxSubLayersInLayerSetMinus1**[**OlsIdxToLsIdx**[i]], inclusive, due to **sub_layer_dpb_info_present_flag**[i][j] being equal to 0, it is inferred to be equal to **max_vps_num_reorder_pics**[i][j - 1].

max_vps_latency_increase_plus1[i][j] not equal to 0 is used to compute the value of **MaxVpsLatencyPictures**[i][j], which, when **HighestTid** is equal to j, specifies the maximum number of access units containing a picture with **PicOutputFlag** equal to 1 in the i-th OLS that can precede any access unit auA that contains a picture with **PicOutputFlag** equal to 1 in the CVS in output order and follow the access unit auA that contains a picture with **PicOutputFlag** equal to 1 in decoding order. When **max_vps_latency_increase_plus1**[i][j] is not present for j in the range of 1 to **MaxSubLayersInLayerSetMinus1**[**OlsIdxToLsIdx**[i]], inclusive, due to **sub_layer_dpb_info_present_flag**[i][j] being equal to 0, it is inferred to be equal to **max_vps_latency_increase_plus1**[i][j - 1].

When **max_vps_latency_increase_plus1**[i][j] is not equal to 0, the value of **MaxVpsLatencyPictures**[i][j] is specified as follows:

$$\text{MaxVpsLatencyPictures}[i][j] = \text{max_vps_num_reorder_pics}[i][j] + \text{max_vps_latency_increase_plus1}[i][j] - 1$$

(F-15)

When **max_vps_latency_increase_plus1**[i][j] is equal to 0, no corresponding limit is expressed. The value of **max_vps_latency_increase_plus1**[i][j] shall be in the range of 0 to $2^{32} - 2$, inclusive.

F.7.4.3.1.4 VPS VUI semantics

NOTE 1 – When **vps_vui_present_flag** is equal to 0, some of the VPS VUI flags, such as **cross_layer_pic_type_aligned_flag**, **tiles_not_in_use_flag** and **wpp_not_in_use_flag**, are not present and no value is inferred for them. In this case, the semantics of these flags are unspecified and it should be interpreted as such that it is unknown whether the constraints associated with these flags being equal to 1 apply or not; in other words, in this case those constraints may or may not apply.

cross_layer_pic_type_aligned_flag equal to 1 specifies that within a CVS that refers to the VPS, all VCL NAL units that belong to an access unit have the same value of **nal_unit_type**. **cross_layer_pic_type_aligned_flag** equal to 0 specifies that within a CVS that refers to the VPS, all VCL NAL units in each access unit may or may not have the same value of **nal_unit_type**.

cross_layer_irap_aligned_flag equal to 1 specifies that IRAP pictures in the CVS are cross-layer aligned, i.e., when a picture pictureA of a layer layerA in an access unit is an IRAP picture, each picture pictureB in the same access unit that

belongs to a direct reference layer of layerA or that belongs to a layer for which layerA is a direct reference layer of that layer is an IRAP picture and the VCL NAL units of pictureB have the same value of nal_unit_type as that of pictureA. cross_layer_irap_aligned_flag equal to 0 specifies that the above restriction may or may not apply. When not present, the value of cross_layer_irap_aligned_flag is inferred to be equal to vps_vui_present_flag.

all_layers_idr_aligned_flag equal to 1 indicates that within each access unit for which the VCL NAL units refer to the VPS, when one picture is an IRAP picture, all the pictures in the same access unit are IDR pictures and have the same value of nal_unit_type. all_layers_idr_aligned_flag equal to 0 specifies that the above restriction may or may not apply. When not present, the value of all_layers_idr_aligned_flag is inferred to be equal to 0.

bit_rate_present_vps_flag equal to 1 specifies that the syntax element bit_rate_present_flag[i][j] is present. bit_rate_present_vps_flag equal to 0 specifies that the syntax element bit_rate_present_flag[i][j] is not present.

pic_rate_present_vps_flag equal to 1 specifies that the syntax element pic_rate_present_flag[i][j] is present. pic_rate_present_vps_flag equal to 0 specifies that the syntax element pic_rate_present_flag[i][j] is not present.

bit_rate_present_flag[i][j] equal to 1 specifies that the bit rate information for the j-th subset of the i-th layer set is present. bit_rate_present_flag[i] equal to 0 specifies that the bit rate information for the j-th subset of the i-th layer set is not present. The j-th subset of a layer set is derived as follows:

- If i is less than or equal to vps_num_layer_sets_minus1, the j-th subset of a layer set is the output of the sub-bitstream extraction process specified in clause F.10.1, when it is invoked with inBitstream equal to the layer set, tIdTarget equal to j and layerIdListTarget equal to the layer identifier list associated with the layer set as inputs.
- Otherwise (i is greater than vps_num_layer_sets_minus1), the j-th subset of a layer set is the output of the sub-bitstream extraction process specified in clause F.10.3, when it is invoked with inBitstream equal to the layer set, tIdTarget equal to j and layerIdListTarget equal to the layer identifier list associated with the layer set as inputs.

When not present, the value of bit_rate_present_flag[i][j] is inferred to be equal to 0.

pic_rate_present_flag[i][j] equal to 1 specifies that picture rate information for the j-th subset of the i-th layer set is present. pic_rate_present_flag[i][j] equal to 0 specifies that picture rate information for the j-th subset of the i-th layer set is not present. When not present, the value of pic_rate_present_flag[i][j] is inferred to be equal to 0.

avg_bit_rate[i][j] indicates the average bit rate of the j-th subset of the i-th layer set, in bits per second. The value is given by BitRateBPS(avg_bit_rate[i][j]) with the function BitRateBPS() being specified as follows:

$$\text{BitRateBPS}(x) = (x \& (2^{14} - 1)) * 10^{(2 + (x \gg 14))} \quad (\text{F-16})$$

The average bit rate is derived according to the access unit removal time specified in clause F.13. In the following, bTotal is the number of bits in all NAL units of the j-th subset of the i-th layer set, t₁ is the removal time (in seconds) of the first access unit to which the VPS applies and t₂ is the removal time (in seconds) of the last access unit (in decoding order) to which the VPS applies. With x specifying the value of avg_bit_rate[i][j], the following applies:

- If t₁ is not equal to t₂, the following condition shall be true:

$$(x \& (2^{14} - 1)) = \text{Round}(bTotal \div ((t_2 - t_1) * 10^{(2 + (x \gg 14))})) \quad (\text{F-17})$$

- Otherwise (t₁ is equal to t₂), the following condition shall be true:

$$(x \& (2^{14} - 1)) = 0 \quad (\text{F-18})$$

max_bit_rate_layer[i][j] indicates an upper bound for the bit rate of the j-th subset of the i-th layer set in any one-second time window of access unit removal time as specified in clause F.13. The upper bound for the bit rate in bits per second is given by BitRateBPS(max_bit_rate_layer[i][j]). The bit rate values are derived according to the access unit removal time specified in clause F.13. In the following, t₁ is any point in time (in seconds), t₂ is set equal to t₁ + 1 ÷ 100 and bTotal is the number of bits in all NAL units of access units with a removal time greater than or equal to t₁ and less than t₂. With x specifying the value of max_bit_rate_layer[i][j], the following condition shall be obeyed for all values of t₁:

$$(x \& (2^{14} - 1)) \geq bTotal \div ((t_2 - t_1) * 10^{(2 + (x \gg 14))}) \quad (\text{F-19})$$

constant_pic_rate_idc[i][j] indicates whether the picture rate of the j-th subset of the i-th layer set is constant. In the following, a temporal segment tSeg is any set of two or more consecutive access units, in decoding order, of the j-th subset of the i-th layer set, auTotal(tSeg) is the number of access units in the temporal segment tSeg, t₁(tSeg) is the removal time (in seconds) of the first access unit (in decoding order) of the temporal segment tSeg, t₂(tSeg) is the removal time (in seconds) of the last access unit (in decoding order) of the temporal segment tSeg, and avgPicRate(tSeg) is the average picture rate in the temporal segment tSeg, and is specified as follows:

$$\text{avgPicRate}(tSeg) = \text{Round}(auTotal(tSeg) * 256 \div (t_2(tSeg) - t_1(tSeg))) \quad (\text{F-20})$$

If the j -th subset of the i -th layer set only contains one or two access units or the value of $\text{avgPicRate}(t_{\text{Seg}})$ is constant over all the temporal segments, the picture rate is constant; otherwise, the picture rate is not constant.

$\text{constant_pic_rate_idc}[i][j]$ equal to 0 indicates that the picture rate of the j -th subset of the i -th layer set is not constant. $\text{constant_pic_rate_idc}[i][j]$ equal to 1 indicates that the picture rate of the j -th subset of the i -th layer set is constant. $\text{constant_pic_rate_idc}[i][j]$ equal to 2 indicates that the picture rate of the j -th subset of the i -th layer set may or may not be constant. The value of $\text{constant_pic_rate_idc}[i][j]$ shall be in the range of 0 to 2, inclusive.

$\text{avg_pic_rate}[i]$ indicates the average picture rate, in units of picture per 256 seconds, of the j -th subset of the layer set. With auTotal being the number of access units in the j -th subset of the i -th layer set, t_1 being the removal time (in seconds) of the first access unit to which the VPS applies, and t_2 being the removal time (in seconds) of the last access unit (in decoding order) to which the VPS applies, the following applies:

- If t_1 is not equal to t_2 , the following condition shall be true:

$$\text{avg_pic_rate}[i] == \text{Round}(\text{auTotal} * 256 \div (t_2 - t_1)) \quad (\text{F-21})$$

- Otherwise (t_1 is equal to t_2), the following condition shall be true:

$$\text{avg_pic_rate}[i] == 0 \quad (\text{F-22})$$

video_signal_info_idx_present_flag equal to 1 specifies that the syntax elements $\text{vps_num_video_signal_info_minus1}$ is present and the syntax element $\text{vps_video_signal_info_idx}[i]$ may be present. **video_signal_info_idx_present_flag** equal to 0 specifies that the syntax elements $\text{vps_num_video_signal_info_minus1}$ and $\text{vps_video_signal_info_idx}[i]$ are not present.

vps_num_video_signal_info_minus1 plus 1 specifies the number of the following $\text{video_signal_info}()$ syntax structures in the VPS. When not present, the value of $\text{vps_num_video_signal_info_minus1}$ is inferred to be equal to $\text{MaxLayersMinus1} - (\text{vps_base_layer_internal_flag} ? 0 : 1)$.

vps_video_signal_info_idx $[i]$ specifies the index, into the list of $\text{video_signal_info}()$ syntax structures in the VPS, of the $\text{video_signal_info}()$ syntax structure that applies to the layer with nuh_layer_id equal to $\text{layer_id_in_nuh}[i]$. If **video_signal_info_idx_present_flag** is equal to 0, the value of $\text{vps_video_signal_info_idx}[i]$ for each value of i in the range of $(\text{vps_base_layer_internal_flag} ? 0 : 1)$ to MaxLayersMinus1 , inclusive, is inferred to be equal to i . Otherwise, when $\text{vps_num_video_signal_info_minus1}$ is equal to 0, the value of $\text{vps_video_signal_info_idx}[i]$ for each value of i in the range of $\text{vps_base_layer_internal_flag}$ to MaxLayersMinus1 , inclusive, is inferred to be equal to 0. The value of $\text{vps_video_signal_info_idx}[i]$ for each value of i in the range of $(\text{vps_base_layer_internal_flag} ? 0 : 1)$ to MaxLayersMinus1 , inclusive, shall be in the range of 0 to $\text{vps_num_video_signal_info_minus1}$, inclusive.

tiles_not_in_use_flag equal to 1 indicates that the value of **tiles_enabled_flag** is equal to 0 for each PPS that is referred to by at least one picture referring to the VPS. **tiles_not_in_use_flag** equal to 0 indicates that such a restriction may or may not apply.

tiles_in_use_flag $[i]$ equal to 1 indicates that the value of **tiles_enabled_flag** is equal to 1 for each PPS that is referred to by at least one picture of the i -th layer specified by the VPS. **tiles_in_use_flag** $[i]$ equal to 0 indicates that such a restriction may or may not apply. When not present, the value of **tiles_in_use_flag** $[i]$ is inferred to be equal to 0.

loop_filter_not_across_tiles_flag $[i]$ equal to 1 indicates that the value of **loop_filter_across_tiles_enabled_flag** is equal to 0 for each PPS that is referred to by at least one picture of the i -th layer specified by the VPS. **loop_filter_not_across_tiles_flag** $[i]$ equal to 0 indicates that such a restriction may or may not apply. When not present, the value of **loop_filter_not_across_tiles_flag** $[i]$ is inferred to be equal to 0.

tile_boundaries_aligned_flag $[i][j]$ equal to 1 indicates that, when any two samples of one picture of the i -th layer specified by the VPS belong to one tile, the two collocated samples, when both present in the picture of the j -th direct reference layer of the i -th layer, belong to one tile, and when any two samples of one picture of the i -th layer belong to different tiles, the two collocated samples, when both present in the picture of the j -th direct reference layer of the i -th layer belong to different tiles. **tile_boundaries_aligned_flag** equal to 0 indicates that such a restriction may or may not apply. When not present, the value of **tile_boundaries_aligned_flag** $[i][j]$ is inferred to be equal to 0.

wpp_not_in_use_flag equal to 1 indicates that the value of **entropy_coding_sync_enabled_flag** is equal to 0 for each PPS that is referred to by at least one picture referring to the VPS. **wpp_not_in_use_flag** equal to 0 indicates that such a restriction may or may not apply.

wpp_in_use_flag $[i]$ equal to 1 indicates that the value of **entropy_coding_sync_enabled_flag** is equal to 1 for each PPS that is referred to by at least one picture of the i -th layer specified by the VPS. **wpp_in_use_flag** $[i]$ equal to 0 indicates that such a restriction may or may not apply. When not present, the value of **wpp_in_use_flag** $[i]$ is inferred to be equal to 0.

single_layer_for_non_irap_flag equal to 1 indicates the following:

- If `vps_base_layer_internal_flag` is equal to 1, `single_layer_for_non_irap_flag` equal to 1 indicates that either one of the following is true for each access unit for which this VPS is the active VPS:
 - All the VCL NAL units of an access unit have the same `nuh_layer_id` value.
 - Two `nuh_layer_id` values are used by the VCL NAL units of an access unit and the picture with the greater `nuh_layer_id` value is an IRAP picture.
- Otherwise (`vps_base_layer_internal_flag` is equal to 0), `single_layer_for_non_irap_flag` equal to 1 indicates that any one of the following is true for each access unit for which this VPS is the active VPS:
 - The decoded picture with `nuh_layer_id` equal to 0 is not provided for the access unit by external means and the access unit contains one coded picture.
 - The decoded picture with `nuh_layer_id` equal to 0 is not provided for the access unit by external means, the access unit contains two coded pictures and the picture with the greater `nuh_layer_id` value is an IRAP picture.
 - The decoded picture with `nuh_layer_id` equal to 0 is provided for an access unit by external means and the access unit contains one coded picture that is an IRAP picture.

`single_layer_for_non_irap_flag` equal to 0 indicates that the above constraints may or may not apply. When not present, the value of `single_layer_for_non_irap_flag` is inferred to be equal to 0.

higher_layer_irap_skip_flag equal to 1 indicates that each IRAP picture `currIrapPic` is constrained as specified below. `currIrapPic` is derived as follows for each access unit `currAu` for which this VPS is the active VPS:

- If `vps_base_layer_internal_flag` is equal to 1, `currAu` contains two coded pictures and the picture with the greater `nuh_layer_id` value is an IRAP picture, let `currIrapPic` be that IRAP picture.
- Otherwise, if `vps_base_layer_internal_flag` is equal to 0, a decoded picture with `nuh_layer_id` equal to 0 is not provided for `currAu` by external means, `currAu` contains two coded pictures and the picture with the greater `nuh_layer_id` value is an IRAP picture, let `currIrapPic` be that IRAP picture.
- Otherwise, if `vps_base_layer_internal_flag` is equal to 0, the decoded picture with `nuh_layer_id` equal to 0 is provided for `currAu` by external means and the access unit contains one coded picture that is an IRAP picture, let `currIrapPic` be that IRAP picture.
- Otherwise, `currIrapPic` is not derived for `currAu`.

The following constraints apply for each picture `currIrapPic`:

- For all slices of the IRAP picture:
 - `slice_type` shall be equal to P.
 - `slice_sao_luma_flag` and `slice_sao_chroma_flag` shall both be equal to 0.
 - `five_minus_max_num_merge_cand` shall be equal to 4.
 - `weighted_pred_flag` shall be equal to 0 in the PPS that is referred to by the slices.
- For all coding units of the IRAP picture:
 - `cu_skip_flag[i][j]` shall be equal to 1.

When `single_layer_for_non_irap_flag` is equal to 0, `higher_layer_irap_skip_flag` shall be equal to 0. When `higher_layer_irap_skip_flag` is not present it is inferred to be equal to 0.

NOTE 2 – When `vps_base_layer_internal_flag` is equal to 1, an encoder may set `single_layer_for_non_irap_flag` equal to 1 as an indication to a decoder that at most two pictures are present in any access unit and whenever there are two pictures in the same access unit, the one with the higher value of `nuh_layer_id` is an IRAP picture. The encoder may additionally set `higher_layer_irap_skip_flag` equal to 1 as an indication to a decoder that whenever there are two pictures in the same access unit, the one with the higher value of `nuh_layer_id` is an IRAP picture for which the decoded samples can be derived by applying the inter-layer reference picture derivation process specified in clause H.8.1.4 with the other picture with the lower value of `nuh_layer_id` as input.

ilp_restricted_ref_layers_flag equal to 1 indicates that additional restrictions on inter-layer prediction as specified below apply for each direct reference layer of each layer specified by the VPS. `ilp_restricted_ref_layers_flag` equal to 0 indicates that additional restrictions on inter-layer prediction may or may not apply. When not present, the value of `ilp_restricted_ref_layers_flag` is inferred to be equal to 0.

min_spatial_segment_offset_plus1[i][j] indicates the spatial region, in each picture of the j-th direct reference layer of the i-th layer, that is not used for inter-layer prediction for decoding of any picture of the i-th layer, by itself or together with `min_horizontal_ctu_offset_plus1[i][j]`, as specified below.

The variables `refLayerPicWidthInCtbsY[i][j]` and `refLayerPicHeightInCtbsY[i][j]` are set equal to `PicWidthInCtbsY` and `PicHeightInCtbsY`, respectively, of the j-th direct reference layer of the i-th layer.

The value of $\text{min_spatial_segment_offset_plus1}[i][j]$ shall be in the range of 0 to $\text{refLayerPicWidthInCtbsY}[i][j] * \text{refLayerPicHeightInCtbsY}[i][j]$, inclusive. When not present, the value of $\text{min_spatial_segment_offset_plus1}[i][j]$ is inferred to be equal to 0.

ctu_based_offset_enabled_flag $[i][j]$ equal to 1 specifies that the spatial region, in units of CTUs, in each picture of the j -th direct reference layer of the i -th layer, that is not used for inter-layer prediction for decoding of any picture of the i -th layer is indicated by $\text{min_spatial_segment_offset_plus1}[i][j]$ and $\text{min_horizontal_ctu_offset_plus1}[i][j]$ together. $\text{ctu_based_offset_enabled_flag}[i][j]$ equal to 0 specifies that the spatial region, in units of slice segments, tiles or CTU rows, in each picture of the j -th direct reference layer of the i -th layer, that is not used for inter-layer prediction for decoding of any picture of the i -th layer is indicated by $\text{min_spatial_segment_offset_plus1}[i]$ only. When not present, the value of $\text{ctu_based_offset_enabled_flag}[i]$ is inferred to be equal to 0.

min_horizontal_ctu_offset_plus1 $[i][j]$, when $\text{ctu_based_offset_enabled_flag}[i][j]$ is equal to 1, indicates the spatial region, in each picture of the j -th direct reference layer of the i -th layer, that is not used for inter-layer prediction for decoding of any picture of the i -th layer, together with $\text{min_spatial_segment_offset_plus1}[i][j]$, as specified below. The value of $\text{min_horizontal_ctu_offset_plus1}[i][j]$ shall be in the range of 0 to $\text{refLayerPicWidthInCtbsY}[i][j]$, inclusive.

When $\text{ctu_based_offset_enabled_flag}[i][j]$ is equal to 1, the variable $\text{minHorizontalCtbOffset}[i][j]$ is derived as follows:

$$\text{minHorizontalCtbOffset}[i][j] = (\text{min_horizontal_ctu_offset_plus1}[i][j] > 0) ? \quad (\text{F-23})$$

$$(\text{min_horizontal_ctu_offset_plus1}[i][j] - 1) : (\text{refLayerPicWidthInCtbsY}[i][j] - 1)$$

For any i in the range of 1 to MaxLayersMinus1 , inclusive, and any j in the range of 0 to $\text{NumDirectRefLayers}[\text{layer_id_in_nuh}[i]]$, inclusive, when $\text{min_spatial_segment_offset_plus1}[i][j]$ is greater than 0, it is a requirement of bitstream conformance that the following shall apply:

- Let picA be a picture in the i -th layer and let picB be the direct reference layer picture of picA with nuh_layer_id equal to $\text{IdDirectRefLayer}[\text{layer_id_in_nuh}[i]][j]$. The variable $\text{colCtbAddr}[i][j]$ that denotes the raster scan address of the collocated CTU, in the direct reference layer picture picB , of the CTU with raster scan address equal to $\text{ctbAddr}[i]$ in picA is derived as specified in the following:
 - The variables curPicWidthY , curPicHeightY , curCtbLog2SizeY and $\text{curPicWidthInCtbsY}$ are set equal to $\text{pic_width_in_luma_samples}$, $\text{pic_height_in_luma_samples}$, CtbLog2SizeY and PicWidthInCtbsY , respectively, of the i -th layer picture picA .
 - The variables refPicWidthY , refPicHeightY , refCtbLog2SizeY and $\text{refPicWidthInCtbsY}$ are set equal to $\text{pic_width_in_luma_samples}$, $\text{pic_height_in_luma_samples}$, CtbLog2SizeY and PicWidthInCtbsY , respectively, of the direct reference layer picture picB .
 - If $\text{scaled_ref_layer_offset_present_flag}[\text{IdDirectRefLayer}[\text{layer_id_in_nuh}[i]][j]]$ in the PPS referred to by the i -th layer picture picA is equal to 1, the variables scaledLeftOffset , scaledTopOffset , scaledRightOffset and $\text{scaledBottomOffset}$ are set equal to the left scaled reference layer offset $\text{ScaledRefLayerLeftOffset}$, the top scaled reference layer offset $\text{ScaledRefLayerTopOffset}$, the right scaled reference layer offset $\text{ScaledRefLayerRightOffset}$ and the bottom scaled reference layer offset $\text{ScaledRefLayerBottomOffset}$, respectively, for the direct reference layer picture picB of the i -th layer picture picA .
 - Otherwise ($\text{scaled_ref_layer_offset_present_flag}[\text{IdDirectRefLayer}[\text{layer_id_in_nuh}[i]][j]]$ is equal to 0), the variables scaledLeftOffset , scaledTopOffset , scaledRightOffset and $\text{scaledBottomOffset}$ are set equal to 0.
 - If $\text{ref_region_offset_present_flag}[\text{IdDirectRefLayer}[\text{layer_id_in_nuh}[i]][j]]$ in the PPS referred to by the i -th layer picture picA is equal to 1, the variables $\text{refRegionLeftOffset}$, $\text{refRegionTopOffset}$, $\text{refRegionRightOffset}$ and $\text{refRegionBottomOffset}$ are set equal to the left reference region offset $\text{RefLayerRegionLeftOffset}$, the top reference region offset $\text{RefLayerRegionTopOffset}$, the right reference region offset $\text{RefLayerRegionRightOffset}$ and the bottom reference region offset $\text{RefLayerRegionBottomOffset}$, respectively, for the direct reference layer picture picB of the i -th layer picture picA .
 - Otherwise ($\text{ref_region_offset_present_flag}[\text{IdDirectRefLayer}[\text{layer_id_in_nuh}[i]][j]]$ is equal to 0), the variables $\text{refRegionLeftOffset}$, $\text{refRegionTopOffset}$, $\text{refRegionRightOffset}$ and $\text{refRegionBottomOffset}$ are set equal to 0.
 - The variables (xP, yP) specifying the location of the top-left luma sample of the CTU with raster scan address equal to ctbAddr relative to top-left luma sample in picA are derived as follows:

$$xP = (\text{ctbAddr}[i] \% \text{curPicWidthInCtbsY}) \ll \text{curCtbLog2SizeY} \quad (\text{F-24})$$

$$yP = (\text{ctbAddr}[i] / \text{curPicWidthInCtbsY}) \ll \text{curCtbLog2SizeY} \quad (\text{F-25})$$

- The variables (xPCol, yPCol) specifying the collocated luma sample location in picB of the luma sample location (xP, yP) in picA are derived as follows:

$$\text{refRegionWidth} = \text{refPicWidthY} - \text{refRegionLeftOffset} - \text{refRegionRightOffset} \quad (\text{F-26})$$

$$\text{refRegionHeight} = \text{refPicHeightY} - \text{refRegionTopOffset} - \text{refRegionBottomOffset} \quad (\text{F-27})$$

$$\text{scaledRefRegionWidth} = \text{curPicHeightY} - \text{scaledLeftOffset} - \text{scaledRightOffset} \quad (\text{F-28})$$

$$\text{scaledRefRegionHeight} = \text{curPicHeightY} - \text{scaledTopOffset} - \text{scaledBottomOffset} \quad (\text{F-29})$$

$$\text{scaleX} = ((\text{refRegionWidth} \ll 16) + (\text{scaledRefRegionWidth} \gg 1)) / \text{scaledRefRegionWidth} \quad (\text{F-30})$$

$$\text{scaleY} = ((\text{refRegionHeight} \ll 16) + (\text{scaledRefRegionHeight} \gg 1)) / \text{scaledRefRegionHeight} \quad (\text{F-31})$$

$$\text{xPCol} = \text{Clip3}(0, (\text{refPicWidthY} - 1), \quad (\text{F-32})$$

$$((((\text{xP} - \text{scaledLeftOffset}) * \text{scaleX} + (1 \ll 15)) \gg 16) + \text{refRegionLeftOffset}))$$

$$\text{yPCol} = \text{Clip3}(0, (\text{refPicHeightY} - 1), \quad (\text{F-33})$$

$$((((\text{yP} - \text{scaledTopOffset}) * \text{scaleY} + (1 \ll 15)) \gg 16) + \text{refRegionTopOffset}))$$

- The variable colCtbAddr[i][j] is derived as follows:

$$\text{xColCtb}[i][j] = \text{xPCol} \gg \text{refCtbLog2SizeY} \quad (\text{F-34})$$

$$\text{yColCtb}[i][j] = \text{yPCol} \gg \text{refCtbLog2SizeY} \quad (\text{F-35})$$

$$\text{colCtbAddr}[i][j] = \text{xColCtb}[i][j] + (\text{yColCtb}[i][j] * \text{refPicWidthInCtbsY}) \quad (\text{F-36})$$

- If ctu_based_offset_enabled_flag[i][j] is equal to 0, exactly one of the following applies:
 - In each PPS referred to by a picture in the j-th direct reference layer of the i-th layer, tiles_enabled_flag is equal to 0 and entropy_coding_sync_enabled_flag is equal to 0 and the following applies:
 - Let slice segment A be any slice segment of a picture of the i-th layer and ctbAddr[i] be the raster scan address of the last CTU in slice segment A. Let slice segment B be the slice segment that belongs to the same access unit as slice segment A, belongs to the j-th direct reference layer of the i-th layer and contains the CTU with raster scan address colCtbAddr[i][j]. Let slice segment C be the slice segment that is in the same picture as slice segment B and follows slice segment B in decoding order, and between slice segment B and slice segment C there are min_spatial_segment_offset_plus1[i] – 1 slice segments in decoding order. When slice segment C is present, the syntax elements of slice segment A are constrained such that no sample or syntax elements values in slice segment C or any slice segment of the same picture following slice segment C in decoding order are used for inter-layer prediction in the decoding process of any samples within slice segment A.
 - In each PPS referred to by a picture in the j-th direct reference layer of the i-th layer, tiles_enabled_flag is equal to 1 and entropy_coding_sync_enabled_flag is equal to 0 and the following applies:
 - Let tile A be any tile in any picture picA of the i-th layer and ctbAddr[i] be the raster scan address of the last CTU in tile A. Let tile B be the tile that is in the picture picB belonging to the same access unit as picA and belonging to the j-th direct reference layer of the i-th layer and that contains the CTU with raster scan address colCtbAddr[i][j]. Let tile C be the tile that is also in picB and follows tile B in decoding order, and between tile B and tile C there are min_spatial_segment_offset_plus1[i] – 1 tiles in decoding order. When tile C is present, the syntax elements of tile A are constrained such that no sample or syntax elements values in tile C or any tile of the same picture following tile C in decoding order are used for inter-layer prediction in the decoding process of any samples within tile A.
 - In each PPS referred to by a picture in the j-th direct reference layer of the i-th layer, tiles_enabled_flag is equal to 0 and entropy_coding_sync_enabled_flag is equal to 1 and the following applies:
 - Let CTU row A be any CTU row in any picture picA of the i-th layer and ctbAddr[i] be the raster scan address of the last CTU in CTU row A. Let CTU row B be the CTU row that is in the picture picB belonging to the same access unit as picA and belonging to the j-th direct reference layer of the i-th layer and that contains the CTU with raster scan address colCtbAddr[i][j]. Let CTU row C be the CTU row that is also in picB and follows CTU row B in decoding order and between CTU row B and CTU row C there are

$\text{min_spatial_segment_offset_plus1}[i] - 1$ CTU rows in decoding order. When CTU row C is present, the syntax elements of CTU row A are constrained such that no sample or syntax elements values in CTU row C or row of the same picture following CTU row C are used for inter-layer prediction in the decoding process of any samples within CTU row A.

- Otherwise ($\text{ctu_based_offset_enabled_flag}[i][j]$ is equal to 1), the following applies:

- The variable $\text{refCtbAddr}[i][j]$ is derived as follows:

$$\begin{aligned} \text{xOffset}[i][j] = & \\ & ((\text{xColCtb}[i][j] + \text{minHorizontalCtbOffset}[i][j]) > (\text{refPicWidthInCtbsY}[i][j] - 1)) ? \\ & (\text{refPicWidthInCtbsY}[i][j] - 1 - \text{xColCtb}[i][j]) : \text{minHorizontalCtbOffset}[i][j] \quad (\text{F-37}) \end{aligned}$$

$$\text{yOffset}[i][j] = (\text{min_spatial_segment_offset_plus1}[i][j] - 1) * \text{refPicWidthInCtbsY}[i][j] \quad (\text{F-38})$$

$$\text{refCtbAddr}[i][j] = \text{colCtbAddr}[i][j] + \text{xOffset}[i][j] + \text{yOffset}[i][j] \quad (\text{F-39})$$

- Let CTU A be any CTU in any picture picA of the i -th layer, and $\text{ctbAddr}[i]$ be the raster scan address ctbAddr of CTU A. Let CTU B be a CTU that is in the picture belonging to the same access unit as picA and belonging to the j -th direct reference layer of the i -th layer and that has raster scan address greater than $\text{refCtbAddr}[i][j]$. When CTU B is present, the syntax elements of CTU A are constrained such that no sample or syntax elements values in CTU B are used for inter-layer prediction in the decoding process of any samples within CTU A.

vps_vui_bsp_hrd_present_flag equal to 0 specifies that no bitstream partition HRD parameters are present in the VPS VUI. **vps_vui_bsp_hrd_present_flag** equal to 1 specifies that bitstream partition HRD parameters are present in the VPS VUI. When **vps_timing_info_present_flag** is equal to 0, the value of **vps_vui_bsp_hrd_present_flag** shall be equal to 0.

base_layer_parameter_set_compatibility_flag $[i]$ equal to 1 specifies that the following constraints apply to the layer with nuh_layer_id equal to $\text{layer_id_in_nuh}[i]$. **base_layer_parameter_set_compatibility_flag** $[i]$ equal to 0 specifies that the following constraints may or may not apply to the layer with nuh_layer_id equal to $\text{layer_id_in_nuh}[i]$.

- Each coded slice segment NAL unit with nuh_layer_id value equal to $\text{layer_id_in_nuh}[i]$ referring to the VPS shall refer to a PPS with nuh_layer_id value equal to 0.
- Each coded slice segment NAL unit with nuh_layer_id value equal to $\text{layer_id_in_nuh}[i]$ referring to the VPS shall refer to a SPS with nuh_layer_id value equal to 0.
- The values of **chroma_format_idc**, **separate_colour_plane_flag**, **pic_width_in_luma_samples**, **pic_height_in_luma_samples**, **bit_depth_luma_minus8**, **bit_depth_chroma_minus8**, **conf_win_left_offset**, **conf_win_right_offset**, **conf_win_top_offset** and **conf_win_bottom_offset**, respectively, of the active SPS for the layer with nuh_layer_id equal to $\text{layer_id_in_nuh}[i]$ shall be the same as the values of **chroma_format_vps_idc**, **separate_colour_plane_vps_flag**, **pic_width_vps_in_luma_samples**, **pic_height_vps_in_luma_samples**, **bit_depth_vps_luma_minus8**, **bit_depth_vps_chroma_minus8**, **conf_win_vps_left_offset**, **conf_win_vps_right_offset**, **conf_win_vps_top_offset** and **conf_win_vps_bottom_offset**, respectively, of the $\text{vps_rep_format_idx}[i]$ -th **rep_format()** syntax structure in the active VPS.

F.7.4.3.1.5 Video signal info semantics

video_vps_format, **video_full_range_vps_flag**, **colour_primaries_vps**, **transfer_characteristics_vps** and **matrix_coeffs_vps** are used for inference of the values of the SPS VUI syntax elements **video_format**, **video_full_range_flag**, **colour_primaries**, **transfer_characteristics**, and **matrix_coeffs**, respectively, for each SPS that refers to the VPS.

For each of these syntax elements, all constraints, if any, that apply to the value of the corresponding SPS VUI syntax element also apply.

F.7.4.3.1.6 VPS VUI bitstream partition HRD parameters semantics

vps_num_add_hrd_params specifies the number of additional **hrd_parameters()** syntax structures present in the VPS. The value of **vps_num_add_hrd_params** shall be in the range of 0 to 1 024 – **vps_num_hrd_parameters**, inclusive.

cprms_add_present_flag $[i]$ equal to 1 specifies that the HRD parameters that are common for all sub-layers are present in the i -th **hrd_parameters()** syntax structure. **cprms_add_present_flag** $[i]$ equal to 0 specifies that the HRD parameters that are common for all sub-layers are not present in the i -th **hrd_parameters()** syntax structure and are derived to be the same as the $(i - 1)$ -th **hrd_parameters()** syntax structure. When **vps_num_hrd_parameters** is equal to 0, the value of **cprms_add_present_flag** $[0]$ is inferred to be equal to 1.

num_sub_layer_hrd_minus1 $[i]$ plus 1 specifies the number of sub-layers for which HRD parameters are signalled in

the `i`-th `hrd_parameters()` syntax structure. The value of `num_sub_layer_hrd_minus1[i]` shall be in the range of 0 to `vps_max_sub_layers_minus1`, inclusive.

num_signalled_partitioning_schemes[h] specifies the number of signalled partitioning schemes for the `h`-th output layer set (OLS). The value of `num_signalled_partitioning_schemes[h]` shall be in the range of 0 to 16, inclusive. When `vps_base_layer_internal_flag` is equal to 1, the value of `num_signalled_partitioning_schemes[0]` is inferred to be equal to 0.

num_partitions_in_scheme_minus1[h][j] plus 1 specifies the number of bitstream partitions for the `j`-th partitioning scheme of the `h`-th OLS. The value of `num_partitions_in_scheme_minus1[h][j]` shall be in the range of 0 to `NumLayersInIdList[OlsIdxToLsIdx[h]] - 1`, inclusive. When `vps_base_layer_internal_flag` is equal to 1, the value of `num_partitions_in_scheme_minus1[0][0]` is inferred to be equal to 0. The value of `num_partitions_in_scheme_minus1[h][0]` is inferred to be equal to `NumLayersInIdList[h] - 1`.

NOTE – The 0-th partitioning scheme for each OLS is inferred to contain the number of bitstream partitions to be equal to the number of layers in the OLS and each bitstream partition contains one layer of the OLS.

layer_included_in_partition_flag[h][j][k][r] equal to 1 specifies that the `r`-th layer in the `h`-th OLS is included in the `k`-th bitstream partition of the `j`-th partitioning scheme of the `h`-th OLS. **layer_included_in_partition_flag[h][j][k][r]** equal to 0 specifies that the `r`-th layer in the `h`-th OLS is not included in the `k`-th bitstream partition of the `j`-th partitioning scheme of the `h`-th OLS. When `vps_base_layer_internal_flag` is equal to 1, the value of **layer_included_in_partition_flag[0][0][0][0]** is inferred to be equal to 1. The value of **layer_included_in_partition_flag[h][0][k][r]** for any value of `h` in the range of 1 to `NumOutputLayerSets - 1`, inclusive, is inferred to be equal to (`k == r`).

It is a requirement of bitstream conformance that the following constraints apply:

- For the `j`-th partitioning scheme of the `h`-th OLS, the bitstream partition with index `k1` shall not include reference layers of any layers in the bitstream partition with index `k2` for any values of `k1` and `k2` in the range of 0 to `num_partitions_in_scheme_minus1[h][j]`, inclusive, such that `k2` is less than `k1`.
- When `vps_base_layer_internal_flag` is equal to 0 and **layer_included_in_partition_flag[h][j][k][0]** is equal to 1 for any value of `h` in the range of 1 to `NumOutputLayerSets - 1`, inclusive, any value of `j` in the range of 0 to `num_signalled_partitioning_schemes[h]`, inclusive, and any value of `k` in the range of 0 to `num_partitions_in_scheme_minus1[h][j]`, inclusive, the value of **layer_included_in_partition_flag[h][j][k][r]** for at least one value of `r` in the range of 1 to `NumLayersInIdList[OlsIdxToLsIdx[h]] - 1`, inclusive, shall be equal to 1.
- For each partitioning scheme with index `j` of the `h`-th OLS and for each layer with layer index `r` among the layers in the `h`-th OLS, there exists one and only one value of `k` in the range of 0 to `num_partitions_in_scheme_minus1[h][j]`, inclusive, such that **layer_included_in_partition_flag[h][j][k][r]** is equal to 1.

num_bsp_schedules_minus1[h][i][t] plus 1 specifies the number of delivery schedules specified for bitstream partitions of the `i`-th partitioning scheme of the `h`-th OLS when `HighestTid` is equal to `t`. The value of `num_bsp_schedules_minus1[h][i][t]` shall be in the range of 0 to 31, inclusive.

The variable `BspSchedCnt[h][i][t]` is set equal to `num_bsp_schedules_minus1[h][i][t] + 1`.

bsp_hrd_idx[h][i][t][j][k] specifies the index of the `hrd_parameters()` syntax structure in the VPS for the `j`-th delivery schedule specified for the `k`-th bitstream partition of the `i`-th partitioning scheme for the `h`-th OLS when `HighestTid` is equal to `t`. The length of the `bsp_hrd_idx[h][i][t][j][k]` syntax element is `Ceil(Log2(vps_num_hrd_parameters + vps_num_add_hrd_params))` bits. The value of `bsp_hrd_idx[h][i][t][j][k]` shall be in the range of 0 to `vps_num_hrd_parameters + vps_num_add_hrd_params - 1`, inclusive. When `vps_num_hrd_parameters + vps_num_add_hrd_params` is equal to 1, the value of `bsp_hrd_idx[h][i][t][j][k]` is inferred to be equal to 0.

bsp_sched_idx[h][i][t][j][k] specifies the index of the delivery schedule within the `sub_layer_hrd_parameters(t)` syntax structure of the `hrd_parameters()` syntax structure with the index `bsp_hrd_idx[h][i][t][j][k]`, that is used as the `j`-th delivery schedule specified for the `k`-th bitstream partition of the `i`-th partitioning scheme for the `h`-th OLS when `HighestTid` is equal to `t`. The value of `bsp_sched_idx[h][i][t][j][k]` shall be in the range of 0 to `cpb_cnt_minus1[t]`, inclusive, where `cpb_cnt_minus1[t]` is found in the `hrd_parameters()` syntax structure corresponding to the index `bsp_hrd_idx[h][i][t][j][k]`.

The following applies for any `j` greater than 0, any `h` in the range of 1 to `NumOutputLayerSets - 1`, inclusive, any `i` in the range 0 to `num_signalled_partitioning_schemes[h]`, inclusive, any `t` in the range of 0 to `MaxSubLayersInLayerSetMinus1[OlsIdxToLsIdx[h]]`, inclusive, and any `k` in the range of 0 to `num_partitions_in_scheme_minus1[h][i]`, inclusive:

- For `x` in the range of 0 to 1, inclusive, the following applies:
 - The variable `bsIdx` is set equal to `bsp_sched_idx[h][i][t][j - x][k]`.

- The variables `bitRateValueMinus1[x]`, `bitRateDuValueMinus1[x]`, `cpbSizeValueMinus1[x]` and `cpbSizeDuValueMinus1[x]` are set equal to the values of the syntax elements `bit_rate_value_minus1[bsIdx]`, `bit_rate_du_value_minus1[bsIdx]`, `cpb_size_value_minus1[bsIdx]` and `cpb_size_du_value_minus1[bsIdx]`, respectively, found in the `sub_layer_hrd_parameters(t)` syntax structure within the `hrd_parameters()` structure with index `bsp_hrd_idx[h][i][t][j - x][k]`.
- It is a requirement of bitstream conformance that all of the following conditions shall be true:
 - `bitRateValueMinus1[0]` is greater than `bitRateValueMinus1[1]`.
 - `bitRateDuValueMinus1[0]` is greater than `bitRateDuValueMinus1[1]`.
 - `cpbSizeValueMinus1[0]` is less than or equal to `cpbSizeValueMinus1[1]`.
 - `cpbSizeDuValueMinus1[0]` is less than or equal to `cpbSizeDuValueMinus1[1]`.

The arrays `BpBitRate` and `BpbSize` for bitstream partition buffer (BPB) are derived as follows:

```

for( h = 1; h < NumOutputLayerSets; h++ )
  for( i = 0; i <= num_signalled_partitioning_schemes[ h ]; i++ )
    for( t = 0; t <= MaxSubLayersInLayerSetMinus1[ OlsIdxToLsIdx[ h ] ]; t++ )
      for( j = 0; j <= num_bsp_schedules_minus1[ h ][ i ][ t ]; j++ )
        for( k = 0; k <= num_partitions_in_scheme_minus1[ h ][ i ][ t ][ j ]; k++ ) {
          bsIdx = bsp_sched_idx[ h ][ i ][ t ][ j ][ k ]
          brDu = ( bit_rate_du_value_minus1[ bsIdx ] + 1 ) * 2(6 + bit_rate_scale)
          brPu = ( bit_rate_value_minus1[ bsIdx ] + 1 ) * 2(6 + bit_rate_scale)
          cpbSizeDu = ( cpb_size_du_value_minus1[ bsIdx ] + 1 ) * 2(4 + cpb_size_du_scale)
          cpbSizePu = ( cpb_size_value_minus1[ bsIdx ] + 1 ) * 2(4 + cpb_size_scale)
          BpBitRate[ h ][ i ][ t ][ j ][ k ] = SubPicHrdFlag ? brDu : brPu
          BpbSize[ h ][ i ][ t ][ j ][ k ] = SubPicHrdFlag ? cpbSizeDu : cpbSizePu
        }

```

(F-40)

where the syntax elements `bit_rate_scale`, `cpb_size_du_scale` and `cpb_size_scale` values are found in the `hrd_parameters()` syntax structure corresponding to the index `bsp_hrd_idx[h][i][t][j][k]`, and the syntax elements `bit_rate_du_value_minus1[bsIdx]`, `bit_rate_value_minus1[bsIdx]`, `cpb_size_du_value_minus1[bsIdx]` and `cpb_size_value_minus1[bsIdx]` are found in the `sub_layer_hrd_parameters(t)` syntax structure within that `hrd_parameters()` syntax structure.

F.7.4.3.2 Sequence parameter set RBSP semantics

F.7.4.3.2.1 General sequence parameter set RBSP semantics

The specifications in clause 7.4.3.2.1 apply with the following additions and modifications:

sps_max_sub_layers_minus1 plus 1 specifies the maximum number of temporal sub-layers that may be present in each CVS referring to the SPS. The value of `sps_max_sub_layers_minus1` shall be in the range of 0 to 6, inclusive. The value of `sps_max_sub_layers_minus1` shall be less than or equal to `vps_max_sub_layers_minus1`. When not present, the value of `sps_max_sub_layers_minus1` is inferred to be equal to $(\text{sps_ext_or_max_sub_layers_minus1} == 7) ? \text{vps_max_sub_layers_minus1} : \text{sps_ext_or_max_sub_layers_minus1}$.

sps_ext_or_max_sub_layers_minus1 is used to infer the value of `sps_max_sub_layers_minus1` and to derive the value of `MultiLayerExtSpsFlag`.

It is a requirement of bitstream conformance that the following applies:

- If the SPS is referred to by any current picture that belongs to an independent non-base layer, the value of `MultiLayerExtSpsFlag` derived from the SPS shall be equal to 0.
- Otherwise, the value of `MultiLayerExtSpsFlag` derived from the SPS shall be equal to 1.

sps_temporal_id_nesting_flag, when `sps_max_sub_layers_minus1` is greater than 0, specifies whether inter prediction is additionally restricted for CVSs referring to the SPS. When `vps_temporal_id_nesting_flag` is equal to 1, `sps_temporal_id_nesting_flag` shall be equal to 1. When `sps_max_sub_layers_minus1` is equal to 0, `sps_temporal_id_nesting_flag` shall be equal to 1. When not present, the value of `sps_temporal_id_nesting_flag` is inferred as follows:

- If `sps_max_sub_layers_minus1` is greater than 0, the value of `sps_temporal_id_nesting_flag` is inferred to be equal to `vps_temporal_id_nesting_flag`.

- Otherwise, the value of `sps_temporal_id_nesting_flag` is inferred to be equal to 1.

NOTE 1 – The syntax element `sps_temporal_id_nesting_flag` is used to indicate that temporal up-switching, i.e., switching from decoding up to any TemporalId tIdN to decoding up to any TemporalId tIdM that is greater than tIdN, is always possible in the CVS.

update_rep_format_flag equal to 1 specifies that `sps_rep_format_idx` is present and that the `sps_rep_format_idx`-th `rep_format()` syntax structures in the active VPS applies to the layers that refer to this SPS. `update_rep_format_flag` equal to 0 specifies that `sps_rep_format_idx` is not present. When the value of `vps_num_rep_formats_minus1` in the active VPS is equal to 0, it is a requirement of bitstream conformance that the value of `update_rep_format_flag` shall be equal to 0. When not present, the value of `update_rep_format_flag` is inferred to be equal to 0.

sps_rep_format_idx specifies the index, into the list of `rep_format()` syntax structures in the VPS, of the `rep_format()` syntax structure that applies to the layers that refer to this SPS. The value of `sps_rep_format_idx` shall be in the range of 0 to `vps_num_rep_formats_minus1`, inclusive.

When a current picture with `nuh_layer_id` `layerIdCurr` greater than 0 refers to an SPS and the layer with `nuh_layer_id` equal to `layerIdCurr` is not an independent non-base layer, the values of `chroma_format_idc`, `separate_colour_plane_flag`, `pic_width_in_luma_samples`, `pic_height_in_luma_samples`, `bit_depth_luma_minus8`, `bit_depth_chroma_minus8`, `conf_win_left_offset`, `conf_win_right_offset`, `conf_win_top_offset` and `conf_win_bottom_offset` are inferred or constrained as follows:

- The variable `repFormatIdx` is derived as follows:
 - If `update_rep_format_flag` is equal to 0, the variable `repFormatIdx` is set equal to `vps_rep_format_idx[LayerIdxInVps[layerIdCurr]]`.
 - Otherwise (`update_rep_format_flag` is equal to 1), `repFormatIdx` is set equal to `sps_rep_format_idx`.
- If `MultiLayerExtSpsFlag` derived from the active SPS for the layer with `nuh_layer_id` equal to `layerIdCurr` is equal to 0, the values of `chroma_format_idc`, `separate_colour_plane_flag`, `pic_width_in_luma_samples`, `pic_height_in_luma_samples`, `bit_depth_luma_minus8`, `bit_depth_chroma_minus8`, `conf_win_left_offset`, `conf_win_right_offset`, `conf_win_top_offset` and `conf_win_bottom_offset`, are inferred to be equal to `chroma_format_vps_idc`, `separate_colour_plane_vps_flag`, `pic_width_vps_in_luma_samples`, `pic_height_vps_in_luma_samples`, `bit_depth_vps_luma_minus8`, `bit_depth_vps_chroma_minus8`, `conf_win_vps_left_offset`, `conf_win_vps_right_offset`, `conf_win_vps_top_offset` and `conf_win_vps_bottom_offset`, respectively, of the `repFormatIdx`-th `rep_format()` syntax structure in the active VPS and the values of `chroma_format_idc`, `separate_colour_plane_flag`, `pic_width_in_luma_samples`, `pic_height_in_luma_samples`, `bit_depth_luma_minus8`, `bit_depth_chroma_minus8`, `conf_win_left_offset`, `conf_win_right_offset`, `conf_win_top_offset` and `conf_win_bottom_offset` signalled in the active SPS for the layer with `nuh_layer_id` equal to `layerIdCurr` are ignored.

NOTE 2 – The values are inferred from the VPS when a layer that is not an independent layer refers to an SPS that is also referred to by the base layer, in which case the SPS has `nuh_layer_id` equal to 0. For independent layers, the values of these parameters in the active SPS for the base layer apply.

- Otherwise, the values of `chroma_format_idc`, `separate_colour_plane_flag`, `pic_width_in_luma_samples`, `pic_height_in_luma_samples`, `bit_depth_luma_minus8`, `bit_depth_chroma_minus8`, `conf_win_left_offset`, `conf_win_right_offset`, `conf_win_top_offset` and `conf_win_bottom_offset` are inferred to be equal to `chroma_format_vps_idc`, `separate_colour_plane_vps_flag`, `pic_width_vps_in_luma_samples`, `pic_height_vps_in_luma_samples`, `bit_depth_vps_luma_minus8`, `bit_depth_vps_chroma_minus8`, `conf_win_vps_left_offset`, `conf_win_vps_right_offset`, `conf_win_vps_top_offset` and `conf_win_vps_bottom_offset` respectively, of the `repFormatIdx`-th `rep_format()` syntax structure in the active VPS.

It is a requirement of bitstream conformance that, when present, the value of `chroma_format_idc`, `pic_width_in_luma_samples`, `pic_height_in_luma_samples`, `bit_depth_luma_minus8` or `bit_depth_chroma_minus8` shall be less than or equal to `chroma_format_vps_idc`, `pic_width_vps_in_luma_samples`, `pic_height_vps_in_luma_samples`, `bit_depth_vps_luma_minus8`, or `bit_depth_vps_chroma_minus8`, respectively, of the `vps_rep_format_idx[j]`-th `rep_format()` syntax structure in the active VPS, where `j` is equal to `LayerIdxInVps[layerIdCurr]`.

sps_max_dec_pic_buffering_minus1[i] plus 1 specifies the maximum required size of the decoded picture buffer for the CVS in units of picture storage buffers when `HighestTid` is equal to `i`. The value of `sps_max_dec_pic_buffering_minus1[i]` shall be in the range of 0 to `MaxDpbSize - 1`, inclusive, where `MaxDpbSize` is as specified in clause A.4. When `i` is greater than 0, `sps_max_dec_pic_buffering_minus1[i]` shall be greater than or equal to `sps_max_dec_pic_buffering_minus1[i - 1]`. The value of `sps_max_dec_pic_buffering_minus1[i]` shall be less than or equal to `vps_max_dec_pic_buffering_minus1[i]` for each value of `i`. When `sps_max_dec_pic_buffering_minus1[i]` is not present for `i` in the range of 0 to `sps_max_sub_layers_minus1 - 1`, inclusive, due to `sps_sub_layer_ordering_info_present_flag` being equal to 0, it is inferred to be equal to `sps_max_dec_pic_buffering_minus1[sps_max_sub_layers_minus1]`.

When `sps_max_dec_pic_buffering_minus1[i]` is not present for `i` in the range of 0 to `sps_max_sub_layers_minus1`, inclusive, due to `MultiLayerExtSpsFlag` being equal to 1, for a layer that refers to the SPS and has `nuh_layer_id` equal to `currLayerId`, the value of `sps_max_dec_pic_buffering_minus1[i]` is inferred to be equal to `max_vps_dec_pic_buffering_minus1[TargetOlsIdx][layerIdx][i]` of the active VPS, where `layerIdx` is equal to the value such that `LayerSetLayerIdList[TargetDecLayerSetIdx][layerIdx]` is equal to `currLayerId`.

sps_infer_scaling_list_flag equal to 1 specifies that the syntax elements of the scaling list data syntax structure of the SPS are inferred to be equal to those of the SPS that is active for the layer with `nuh_layer_id` equal to `sps_scaling_list_ref_layer_id`. **sps_infer_scaling_list_flag** equal to 0 specifies that the syntax elements of the scaling list data syntax structure are not inferred. When not present, the value of `sps_infer_scaling_list_flag` is inferred to be 0.

sps_scaling_list_ref_layer_id specifies the value of the `nuh_layer_id` of the layer for which the active SPS is associated with the same scaling list data as the current SPS. The value of `sps_scaling_list_ref_layer_id` shall be in the range of 0 to 62, inclusive.

When `vps_base_layer_internal_flag` is equal to 0, it is a requirement of bitstream conformance that the value of `sps_scaling_list_ref_layer_id`, when present, shall be greater than 0.

It is a requirement of bitstream conformance that, when an SPS with `nuh_layer_id` equal to `nuhLayerIdA` is active for a layer with `nuh_layer_id` equal to `nuhLayerIdB` and `sps_infer_scaling_list_flag` in the SPS is equal to 1, `sps_infer_scaling_list_flag` shall be equal to 0 for the SPS that is active for the layer with `nuh_layer_id` equal to `sps_scaling_list_ref_layer_id`.

It is a requirement of bitstream conformance that, when an SPS with `nuh_layer_id` equal to `nuhLayerIdA` is active for a layer with `nuh_layer_id` equal to `nuhLayerIdB` and `sps_infer_scaling_list_flag` in the SPS equal to 1, the layer with `nuh_layer_id` equal to `sps_scaling_list_ref_layer_id` shall be a reference layer of the layer with `nuh_layer_id` equal to `nuhLayerIdB`.

It is a requirement of bitstream conformance that, when an SPS is active for a layer with `sps_infer_scaling_list_flag` in the SPS is equal to 1, `scaling_list_enabled_flag` shall be equal to 1 for the SPS that is active for the layer with `nuh_layer_id` equal to `sps_scaling_list_ref_layer_id`.

F.7.4.3.2.2 Sequence parameter set range extension semantics

The specifications in clause 7.4.3.2.2 apply.

F.7.4.3.2.3 Sequence parameter set screen content coding extension semantics

The specifications in clause 7.4.3.2.3 apply.

F.7.4.3.2.4 Sequence parameter set multilayer extension semantics

inter_view_mv_vert_constraint_flag equal to 1 indicates that vertical component of motion vectors used for inter-layer prediction are constrained in the layers for which this SPS RBSP is the active SPS RBSP. When `inter_view_mv_vert_constraint_flag` is equal to 1, the vertical component of the motion vectors used for inter-layer prediction shall be less than or equal to 56 in units of luma samples. When `inter_view_mv_vert_constraint_flag` is equal to 0, no constraint on the vertical component of the motion vectors used for inter-layer prediction is signalled by this flag. When not present, the value of `inter_view_mv_vert_constraint_flag` is inferred to be equal to 0.

F.7.4.3.3 Picture parameter set RBSP semantics

F.7.4.3.3.1 General picture parameter set RBSP semantics

The specifications in clause 7.4.3.3.1 apply with the replacement of the semantics of `pps_scaling_list_data_present_flag` as follows:

pps_scaling_list_data_present_flag equal to 1 specifies that the scaling list data used for the pictures referring to the PPS are derived based on the scaling lists specified by the active SPS and the scaling lists specified by the PPS. `pps_scaling_list_data_present_flag` equal to 0 specifies that the scaling list data used for the pictures referring to the PPS are inferred to be equal to those specified by the active SPS (when `pps_infer_scaling_list_flag` is equal to 0) or is specified by `pps_scaling_list_ref_layer_id` (when `pps_infer_scaling_list_flag` is equal to 1). When `scaling_list_enabled_flag` is equal to 0, the value of `pps_scaling_list_data_present_flag` shall be equal to 0. When `scaling_list_enabled_flag` is equal to 1, `sps_scaling_list_data_present_flag` is equal to 0, `pps_scaling_list_data_present_flag` is equal to 0 and `pps_infer_scaling_list_flag` is equal to 0, the default scaling list data are used to derive the array `ScalingFactor` as specified in clause 7.4.5.

F.7.4.3.3.2 Picture parameter set range extension semantics

The specifications in clause 7.4.3.3.2 apply.

F.7.4.3.3.3 Picture parameter set screen content coding extension semantics

The specifications in clause 7.4.3.3.3 apply.

F.7.4.3.3.4 Picture parameter set multilayer extension semantics

poc_reset_info_present_flag equal to 0 specifies that the syntax element **poc_reset_idc** is not present in the slice segment headers of the slices referring to the PPS. **poc_reset_info_present_flag** equal to 1 specifies that the syntax element **poc_reset_idc** is present in the slice segment headers of the slices referring to the PPS. When not present, the value of **poc_reset_info_present_flag** is inferred to be equal to 0.

pps_infer_scaling_list_flag equal to 1 specifies that the syntax elements of the scaling list data syntax structure of the PPS are inferred to be equal to those of the PPS that is active for the layer with **nuh_layer_id** equal to **pps_scaling_list_ref_layer_id**. **pps_infer_scaling_list_flag** equal to 0 specifies that the syntax elements of the scaling list data syntax structure of the PPS are not inferred. When **scaling_list_enabled_flag** is equal to 0, **nuh_layer_id** of the layer referring to the PPS is equal to 0 or **pps_scaling_list_data_present_flag** is equal to 1, **pps_infer_scaling_list_flag** shall be equal to 0. When not present, the value of **pps_infer_scaling_list_flag** is inferred to be 0.

pps_scaling_list_ref_layer_id specifies the value of **nuh_layer_id** of the layer for which the active PPS has the same scaling list data as the current PPS.

The value of **pps_scaling_list_ref_layer_id** shall be in the range of 0 to 62, inclusive.

When **vps_base_layer_internal_flag** is equal to 0, it is a requirement of bitstream conformance that **pps_scaling_list_ref_layer_id**, when present, shall be greater than 0. It is a requirement of bitstream conformance that, when a PPS with **nuh_layer_id** equal to **nuhLayerIdA** is active for a layer with **nuh_layer_id** equal to **nuhLayerIdB** and **pps_infer_scaling_list_flag** in the PPS is equal to 1, **pps_infer_scaling_list_flag** shall be equal to 0 for the PPS that is active for the layer with **nuh_layer_id** equal to **pps_scaling_list_ref_layer_id**.

It is a requirement of bitstream conformance that, when a PPS with **nuh_layer_id** equal to **nuhLayerIdA** is active for a layer with **nuh_layer_id** equal to **nuhLayerIdB** and **pps_infer_scaling_list_flag** in the PPS equal to 1, the layer with **nuh_layer_id** equal to **pps_scaling_list_ref_layer_id** shall be a reference layer of the layer with **nuh_layer_id** equal to **nuhLayerIdB**.

It is a requirement of bitstream conformance that, when a PPS is active for a layer with **pps_infer_scaling_list_flag** in the PPS equal to 1, **scaling_list_enabled_flag** shall be equal to 1 for the SPS that is active for the layer with **nuh_layer_id** equal to **pps_scaling_list_ref_layer_id**.

num_ref_loc_offsets specifies the number of reference layer location offsets that are present in the PPS. The value of **num_ref_loc_offsets** shall be in the range of 0 to **vps_max_layers_minus1**, inclusive.

ref_loc_offset_layer_id[i] specifies the **nuh_layer_id** value for which the *i*-th reference layer location offset parameters are specified.

NOTE – **ref_loc_offset_layer_id[i]** need not be among the direct reference layers, for example when the spatial correspondence of an auxiliary picture to its associated primary picture is specified.

The *i*-th reference layer location offset parameters consist of the *i*-th scaled reference layer offset parameters, the *i*-th reference region offset parameters and the *i*-th resampling phase set parameters.

scaled_ref_layer_offset_present_flag[i] equal to 1 specifies that the *i*-th scaled reference layer offset parameters are present in the PPS. **scaled_ref_layer_offset_present_flag[i]** equal to 0 specifies that the *i*-th scaled reference layer offset parameters are not present in the PPS. When not present, the value of **scaled_ref_layer_offset_present_flag[i]** is inferred to be equal to 0.

The *i*-th scaled reference layer offset parameters specify the spatial correspondence of a picture referring to this PPS relative to the reference region in a decoded picture with **nuh_layer_id** equal to **ref_loc_offset_layer_id[i]**.

scaled_ref_layer_left_offset[ref_loc_offset_layer_id[i]] specifies the horizontal offset between the sample in the current picture that is collocated with the top-left luma sample of the reference region in a decoded picture with **nuh_layer_id** equal to **ref_loc_offset_layer_id[i]** and the top-left luma sample of the current picture in units of subWC luma samples, where subWC is equal to the SubWidthC of the picture that refers to this PPS. The value of **scaled_ref_layer_left_offset[ref_loc_offset_layer_id[i]]** shall be in the range of -2^{14} to $2^{14} - 1$, inclusive. When not present, the value of **scaled_ref_layer_left_offset[ref_loc_offset_layer_id[i]]** is inferred to be equal to 0.

scaled_ref_layer_top_offset[ref_loc_offset_layer_id[i]] specifies the vertical offset between the sample in the current picture that is collocated with the top-left luma sample of the reference region in a decoded picture with **nuh_layer_id** equal to **ref_loc_offset_layer_id[i]** and the top-left luma sample of the current picture in units of subHC luma samples, where subHC is equal to the SubHeightC of the picture that refers to this PPS. The value of **scaled_ref_layer_top_offset[ref_loc_offset_layer_id[i]]** shall be in the range of -2^{14} to $2^{14} - 1$, inclusive. When not present, the value of **scaled_ref_layer_top_offset[ref_loc_offset_layer_id[i]]** is inferred to be equal to 0.

scaled_ref_layer_right_offset[ref_loc_offset_layer_id[i]] specifies the horizontal offset between the sample in the current picture that is collocated with the bottom-right luma sample of the reference region in a decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i] and the bottom-right luma sample of the current picture in units of subWC luma samples, where subWC is equal to the SubWidthC of the picture that refers to this PPS. The value of scaled_ref_layer_right_offset[ref_loc_offset_layer_id[i]] shall be in the range of -2^{14} to $2^{14} - 1$, inclusive. When not present, the value of scaled_ref_layer_right_offset[ref_loc_offset_layer_id[i]] is inferred to be equal to 0.

scaled_ref_layer_bottom_offset[ref_loc_offset_layer_id[i]] specifies the vertical offset between the sample in the current picture that is collocated with the bottom-right luma sample of the reference region in a decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i] and the bottom-right luma sample of the current picture in units of subHC luma samples, where subHC is equal to the SubHeightC of the picture that refers to this PPS. The value of scaled_ref_layer_bottom_offset[ref_loc_offset_layer_id[i]] shall be in the range of -2^{14} to $2^{14} - 1$, inclusive. When not present, the value of scaled_ref_layer_bottom_offset[ref_loc_offset_layer_id[i]] is inferred to be equal to 0.

Let currTopLeftSample, currBotRightSample, colRefRegionTopLeftSample and colRefRegionBotRightSample be the top-left luma sample of the current picture, the bottom-right luma sample of the current picture, the sample in the current picture that is collocated with the top-left luma sample of the reference region in a decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i], and the sample in the current picture that is collocated with the bottom-right luma sample of the reference region in the decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i], respectively.

When the value of scaled_ref_layer_left_offset[ref_loc_offset_layer_id[i]] is greater than 0, colRefRegionTopLeftSample is located to the right of currTopLeftSample. When the value of scaled_ref_layer_left_offset[ref_loc_offset_layer_id[i]] is less than 0, colRefRegionTopLeftSample is located to the left of currTopLeftSample.

When the value of scaled_ref_layer_top_offset[ref_loc_offset_layer_id[i]] is greater than 0, colRefRegionTopLeftSample is located below currTopLeftSample. When the value of scaled_ref_layer_top_offset[ref_loc_offset_layer_id[i]] is less than 0, colRefRegionTopLeftSample is located above currTopLeftSample.

When the value of scaled_ref_layer_right_offset[ref_loc_offset_layer_id[i]] is greater than 0, colRefRegionBotRightSample is located to the left of currBotRightSample. When the value of scaled_ref_layer_right_offset[ref_loc_offset_layer_id[i]] is less than 0, colRefRegionTopLeftSample is located to the right of currBotRightSample.

When the value of scaled_ref_layer_bottom_offset[ref_loc_offset_layer_id[i]] is greater than 0, colRefRegionBotRightSample is located above currBotRightSample. When the value of scaled_ref_layer_bottom_offset[ref_loc_offset_layer_id[i]] is less than 0, colRefRegionTopLeftSample is located below currBotRightSample.

ref_region_offset_present_flag[i] equal to 1 specifies that the i-th reference region offset parameters are present in the PPS. ref_region_offset_present_flag[i] equal to 0 specifies that the i-th reference region offset parameters are not present in the PPS. When not present, the value of ref_region_offset_present_flag[i] is inferred to be equal to 0.

The i-th reference region offset parameters specify the spatial correspondence of the reference region in the decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i] relative to the same decoded picture.

ref_region_left_offset[ref_loc_offset_layer_id[i]] specifies the horizontal offset between the top-left luma sample of the reference region in the decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i] and the top-left luma sample of the same decoded picture in units of subWC luma samples, where subWC is equal to the SubWidthC of the layer with nuh_layer_id equal to ref_loc_offset_layer_id[i]. The value of ref_region_left_offset[ref_loc_offset_layer_id[i]] shall be in the range of -2^{14} to $2^{14} - 1$, inclusive. When not present, the value of ref_region_left_offset[ref_loc_offset_layer_id[i]] is inferred to be equal to 0.

ref_region_top_offset[ref_loc_offset_layer_id[i]] specifies the vertical offset between the top-left luma sample of the reference region in the decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i] and the top-left luma sample of the same decoded picture in units of subHC luma samples, where subHC is equal to the SubHeightC of the layer with nuh_layer_id equal to ref_loc_offset_layer_id[i]. The value of ref_region_top_offset[ref_loc_offset_layer_id[i]] shall be in the range of -2^{14} to $2^{14} - 1$, inclusive. When not present, the value of ref_region_top_offset[ref_loc_offset_layer_id[i]] is inferred to be equal to 0.

ref_region_right_offset[ref_loc_offset_layer_id[i]] specifies the horizontal offset between the bottom-right luma sample of the reference region in the decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i] and the bottom-right luma sample of the same decoded picture in units of subWC luma samples, where subWC is equal to the SubWidthC of the layer with nuh_layer_id equal to ref_loc_offset_layer_id[i]. The value of ref_layer_right_offset[ref_loc_offset_layer_id[i]] shall be in the range of -2^{14} to $2^{14} - 1$, inclusive. When not present, the value of ref_region_right_offset[ref_loc_offset_layer_id[i]] is inferred to be equal to 0.

ref_region_bottom_offset[ref_loc_offset_layer_id[i]] specifies the vertical offset between the bottom-right luma sample of the reference region in the decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i] and the bottom-right luma sample of the same decoded picture in units of subHC luma samples, where subHC is equal to the SubHeightC of the layer with nuh_layer_id equal to ref_loc_offset_layer_id[i]. The value of ref_layer_bottom_offset[ref_loc_offset_layer_id[i]] shall be in the range of -2^{14} to $2^{14} - 1$, inclusive. When not present, the value of ref_region_bottom_offset[ref_loc_offset_layer_id[i]] is inferred to be equal to 0.

Let refPicTopLeftSample, refPicBotRightSample, refRegionTopLeftSample and refRegionBotRightSample be the top-left luma sample of the decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i], the bottom-right luma sample of the decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i], the top-left luma sample of the reference region in the decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i] and the bottom-right luma sample of the reference region in the decoded picture with nuh_layer_id equal to ref_loc_offset_layer_id[i], respectively.

When the value of ref_region_left_offset[ref_loc_offset_layer_id[i]] is greater than 0, refRegionTopLeftSample is located to the right of refPicTopLeftSample. When the value of ref_region_left_offset[ref_loc_offset_layer_id[i]] is less than 0, refRegionTopLeftSample is located to the left of refPicTopLeftSample.

When the value of ref_region_top_offset[ref_loc_offset_layer_id[i]] is greater than 0, refRegionTopLeftSample is located below refPicTopLeftSample. When the value of ref_region_top_offset[ref_loc_offset_layer_id[i]] is less than 0, refRegionTopLeftSample is located above refPicTopLeftSample.

When the value of ref_region_right_offset[ref_loc_offset_layer_id[i]] is greater than 0, refRegionBotRightSample is located to the left of refPicBotRightSample. When the value of ref_region_right_offset[ref_loc_offset_layer_id[i]] is less than 0, refRegionBotRightSample is located to the right of refPicBotRightSample.

When the value of ref_region_bottom_offset[ref_loc_offset_layer_id[i]] is greater than 0, refRegionBotRightSample is located above refPicBotRightSample. When the value of ref_region_bottom_offset[ref_loc_offset_layer_id[i]] is less than 0, refRegionBotRightSample is located below refPicBotRightSample.

resample_phase_set_present_flag[i] equal to 1 specifies that the i-th resampling phase set is present in the PPS. resample_phase_set_present_flag[i] equal to 0 specifies that the i-th resampling phase set is not present in the PPS. When not present, the value of resample_phase_set_present_flag[i] is inferred to be equal to 0.

The i-th resampling phase set specifies the phase offsets used in resampling process of the direct reference layer picture with nuh_layer_id equal to ref_loc_offset_layer_id[i]. When the layer specified by ref_loc_offset_layer_id[i] is not a direct reference layer of the current layer, the values of the syntax elements phase_hor_luma[ref_loc_offset_layer_id[i]], phase_ver_luma[ref_loc_offset_layer_id[i]], phase_hor_chroma_plus8[ref_loc_offset_layer_id[i]] and phase_ver_chroma_plus8[ref_loc_offset_layer_id[i]] are unspecified and shall be ignored by decoders.

phase_hor_luma[ref_loc_offset_layer_id[i]] specifies the luma phase shift in the horizontal direction used in the resampling process of the direct reference layer picture with nuh_layer_id equal to ref_loc_offset_layer_id[i]. The value of phase_hor_luma[ref_loc_offset_layer_id[i]] shall be in the range of 0 to 31, inclusive. When not present, the value of phase_hor_luma[ref_loc_offset_layer_id[i]] is inferred to be equal to 0.

phase_ver_luma[ref_loc_offset_layer_id[i]] specifies the luma phase shift in the vertical direction used in the resampling of the direct reference layer picture with nuh_layer_id equal to ref_loc_offset_layer_id[i]. The value of phase_ver_luma[ref_loc_offset_layer_id[i]] shall be in the range of 0 to 31, inclusive. When not present, the value of phase_ver_luma[ref_loc_offset_layer_id[i]] is inferred to be equal to 0.

phase_hor_chroma_plus8[ref_loc_offset_layer_id[i]] minus 8 specifies the chroma phase shift in the horizontal direction used in the resampling of the direct reference layer picture with nuh_layer_id equal to ref_loc_offset_layer_id[i]. The value of phase_hor_chroma_plus8[ref_loc_offset_layer_id[i]] shall be in the range of 0 to 63, inclusive. When not present, the value of phase_hor_chroma_plus8[ref_loc_offset_layer_id[i]] is inferred to be equal to 8.

phase_ver_chroma_plus8[ref_loc_offset_layer_id[i]] minus 8 specifies the chroma phase shift in the vertical direction used in the resampling process of the direct reference layer picture with nuh_layer_id equal to ref_loc_offset_layer_id[i]. The value of phase_ver_chroma_plus8[ref_loc_offset_layer_id[i]] shall be in the range of 0 to 63, inclusive. When not present, the value of phase_ver_chroma_plus8[ref_loc_offset_layer_id[i]] is inferred as follows:

- If chroma_format_idc is equal to 3 (4:4:4 chroma format), the value of phase_ver_chroma_plus8[ref_loc_offset_layer_id[i]] is inferred to be equal to 8.
- Otherwise, the value of phase_ver_chroma_plus8[ref_loc_offset_layer_id[i]] is inferred to be equal to $(4 * \text{scaledRefRegHeight} + \text{refRegHeight} / 2) / \text{refRegHeight} + 4$, where the value of scaledRefRegHeight is equal to the value of ScaledRefRegionHeightInSamplesY derived for the direct reference layer picture with nuh_layer_id equal to ref_loc_offset_layer_id[i] of the picture that refers to this PPS, and the value of refRegHeight is equal to RefLayerRegionHeightInSamplesY derived for the direct reference layer picture with nuh_layer_id equal to ref_loc_offset_layer_id[i] of the picture that refers to this PPS.

colour_mapping_enabled_flag equal to 1 specifies the colour mapping process may be applied to the inter-layer reference pictures for the coded pictures referring to the PPS. **colour_mapping_enabled_flag** equal to 0 specifies that the colour mapping process is not applied to the inter-layer reference pictures for the coded pictures referring to the PPS. When not present, the value of **colour_mapping_enabled_flag** is inferred to be equal to 0.

It is a requirement of bitstream conformance that, when **pps_pic_parameter_set_id** is greater than or equal to 8, **colour_mapping_enabled_flag** shall be equal to 0.

F.7.4.3.3.5 General colour mapping table semantics

num_cm_ref_layers_minus1 specifies the number of the following **cm_ref_layer_id[i]** syntax elements. The value of **num_cm_ref_layers_minus1** shall be in the range of 0 to 61, inclusive.

cm_ref_layer_id[i] specifies the **nuh_layer_id** value of the associated direct reference layer picture for which the colour mapping table is specified.

cm_octant_depth specifies the maximal split depth of the colour mapping table. In bitstreams conforming to this version of this Specification, the value of **cm_octant_depth** shall be in the range of 0 to 1, inclusive. Other values for **cm_octant_depth** are reserved for future use by ITU-T | ISO/IEC.

The variable **OctantNumC** is derived as follows:

$$\text{OctantNumC} = 1 \ll \text{cm_octant_depth} \quad (\text{F-41})$$

cm_y_part_num_log2 specifies the number of partitions of the smallest colour mapping table octant for the luma component. In bitstreams conforming to this version of this Specification, the value of $(\text{cm_y_part_num_log2} + \text{cm_octant_depth})$ shall be in the range of 0 to 3, inclusive. Other values for $(\text{cm_y_part_num_log2} + \text{cm_octant_depth})$ are reserved for future use by ITU-T | ISO/IEC.

The variables **OctantNumY** and **PartNumY** are derived as follows:

$$\text{OctantNumY} = 1 \ll (\text{cm_octant_depth} + \text{cm_y_part_num_log2}) \quad (\text{F-42})$$

$$\text{PartNumY} = 1 \ll \text{cm_y_part_num_log2} \quad (\text{F-43})$$

luma_bit_depth_cm_input_minus8 specifies the sample bit depth of the input luma component of the colour mapping process. The variable **BitDepthCmInputY** is derived as follows:

$$\text{BitDepthCmInputY} = 8 + \text{luma_bit_depth_cm_input_minus8} \quad (\text{F-44})$$

It is a requirement of bitstream conformance that the value of **BitDepthCmInputY** shall be equal to the bit depth of the luma component of any reference layer with **nuh_layer_id** equal to one of the **cm_ref_layer_id[i]**, for all **i** in the range of 0 to **num_cm_ref_layers_minus1**, inclusive.

chroma_bit_depth_cm_input_minus8 specifies the sample bit depth of the input chroma components of the colour mapping process. The variable **BitDepthCmInputC** is derived as follows:

$$\text{BitDepthCmInputC} = 8 + \text{chroma_bit_depth_cm_input_minus8} \quad (\text{F-45})$$

It is a requirement of bitstream conformance that the value of **BitDepthCmInputC** shall be equal to the bit depth of the chroma components of any reference layer with **nuh_layer_id** equal to one of the **cm_ref_layer_id[i]**, for all **i** in the range of 0 to **num_cm_ref_layers_minus1**, inclusive.

luma_bit_depth_cm_output_minus8 specifies the sample bit depth of the output luma component of the colour mapping process. The variable **BitDepthCmOutputY** is derived as follows:

$$\text{BitDepthCmOutputY} = 8 + \text{luma_bit_depth_cm_output_minus8} \quad (\text{F-46})$$

It is a requirement of bitstream conformance that the value of **BitDepthCmOutputY** shall be greater than or equal to **BitDepthCmInputY** and less than or equal to the bit depth of the luma components of any picture that refers to this PPS.

chroma_bit_depth_cm_output_minus8 specifies the sample bit depth of the output chroma components of the colour mapping process. The variable **BitDepthCmOutputC** is derived as follows:

$$\text{BitDepthCmOutputC} = 8 + \text{chroma_bit_depth_cm_output_minus8} \quad (\text{F-47})$$

It is a requirement of bitstream conformance that the value of **BitDepthCmOutputC** shall be greater than or equal to **BitDepthCmInputC** and less than or equal to the bit depth of the chroma components of any picture that refers to this PPS.

cm_res_quant_bits specifies the number of least significant bits to be added to the colour mapping coefficient residual values **res_coeff_q** and **res_coeff_r**.

cm_delta_flc_bits_minus1 specifies the delta value used to derive the number of bits used to code the syntax element **res_coeff_r**. The variable **CMResLSBits** is set equal to $\text{Max}(0, (10 + \text{BitDepthCmInputY} - \text{BitDepthCmOutputY} - \text{cm_res_quant_bits} - (\text{cm_delta_flc_bits_minus1} + 1)))$.

cm_adapt_threshold_u_delta specifies the partitioning threshold of the Cb component used in colour mapping process. When not present, the value of **cm_adapt_threshold_u_delta** is inferred to be equal to 0. The value of **cm_adapt_threshold_u_delta** shall be in the range of $-2^{\text{BitDepthCmInputC}-1}$ to $2^{\text{BitDepthCmInputC}-1}-1$, inclusive.

The variable **CMThreshU** is derived as follows:

$$\text{CMThreshU} = \text{cm_adapt_threshold_u_delta} + (1 \ll (\text{BitDepthCmInputC} - 1)) \quad (\text{F-48})$$

cm_adapt_threshold_v_delta specifies the partitioning threshold of the Cr component used in colour mapping process. When not present, the value of **cm_adapt_threshold_v_delta** is inferred to be equal to 0. The value of **cm_adapt_threshold_v_delta** shall be in the range of $-2^{\text{BitDepthCmInputC}-1}$ to $2^{\text{BitDepthCmInputC}-1}-1$, inclusive.

The variable **CMThreshV** is derived as follows:

$$\text{CMThreshV} = \text{cm_adapt_threshold_v_delta} + (1 \ll (\text{BitDepthCmInputC} - 1)) \quad (\text{F-49})$$

F.7.4.3.3.6 Colour mapping octants semantics

split_octant_flag equal to 1 specifies that the current colour mapping octant is further split into eight octants with half length in each of the three dimensions. **split_octant_flag** equal to 0 specifies that the current colour mapping octant is not further split into eight octants. When not present, the value of **split_octant_flag** is inferred to be equal to 0.

coded_res_flag[**idxShiftY**][**idxCb**][**idxCr**][**j**] equal to 1 specifies that the residuals for the **j**-th colour mapping coefficients of the octant with octant index equal to (**idxShiftY**, **idxCb**, **idxCr**) are present. **coded_res_flag**[**idxShiftY**][**idxCb**][**idxCr**][**j**] equal to 0 specifies that the residuals for the **j**-th colour mapping coefficients of the octant with octant index equal to (**idxShiftY**, **idxCb**, **idxCr**) are not present. When not present, the value of **coded_res_flag**[**idxShiftY**][**idxCb**][**idxCr**][**j**] is inferred to be equal to 0.

res_coeff_q[**idxShiftY**][**idxCb**][**idxCr**][**j**][**c**] specifies the quotient of the residual for the **j**-th colour mapping coefficient of the **c**-th colour component of the octant with octant index equal to (**idxShiftY**, **idxCb**, **idxCr**). When not present, the value of **res_coeff_q**[**idxShiftY**][**idxCb**][**idxCr**][**j**][**c**] is inferred to be equal to 0.

res_coeff_r[**idxShiftY**][**idxCb**][**idxCr**][**j**][**c**] specifies the remainder of the residual for the **j**-th colour mapping coefficient of the **c**-th colour component of the octant with octant index equal to (**idxShiftY**, **idxCb**, **idxCr**). The number of bits used to code **res_coeff_r** is equal to **CMResLSBits**. If **CMResLSBits** is equal to 0, **res_coeff_r** is not present. When not present, the value of **res_coeff_r**[**idxShiftY**][**idxCb**][**idxCr**][**j**][**c**] is inferred to be equal to 0.

res_coeff_s[**idxShiftY**][**idxCb**][**idxCr**][**j**][**c**] specifies the sign of the residual for the **j**-th colour mapping coefficient of the **c**-th colour component of the octant with octant index equal to (**idxShiftY**, **idxCb**, **idxCr**). When not present, the value of **res_coeff_s**[**idxShiftY**][**idxCb**][**idxCr**][**j**][**c**] is inferred to be equal to 0.

The variables **cmResCoeff**[**idxShiftY**][**idxCb**][**idxCr**][**j**][**c**], with **idxShiftY** in the range of 0 to **OctantNumY** - 1, inclusive, **idxCb** and **idxCr** both in the range of 0 to **OctantNumC** - 1, inclusive, **j** in the range of 0 to 3, inclusive, and **c** in the range of 0 to 2, inclusive, are derived as follows:

$$\begin{aligned} \text{cmResCoeff}[\text{idxShiftY}][\text{idxCb}][\text{idxCr}][j][c] = \\ (1 - 2 * \text{res_coeff_s}[\text{idxShiftY}][\text{idxCb}][\text{idxCr}][j][c]) * \\ ((\text{res_coeff_q}[\text{idxShiftY}][\text{idxCb}][\text{idxCr}][j][c] \ll \text{CMResLSBits}) + \\ \text{res_coeff_r}[\text{idxShiftY}][\text{idxCb}][\text{idxCr}][j][c]) \ll \text{cm_res_quant_bits}) \end{aligned} \quad (\text{F-50})$$

The colour mapping coefficients **LutY**[**idxY**][**idxCb**][**idxCr**][**j**], **LutCb**[**idxY**][**idxCb**][**idxCr**][**j**], **LutCr**[**idxY**][**idxCb**][**idxCr**][**j**] with **idxY** in the range of 0 and **OctantNumY** - 1, inclusive, **idxCb** in the range of 0 and **OctantNumC** - 1, inclusive, **idxCr** in the range of 0 and **OctantNumC** - 1, inclusive, and **j** in the range of 0 and 3, inclusive, are derived as follows:

$$\begin{aligned} \text{if}(\text{idxY} == 0) \{ \\ \quad \text{predY}[\text{idxY}][\text{idxCb}][\text{idxCr}][j] = (j == 0) ? 1024 : 0 \\ \quad \text{predCb}[\text{idxY}][\text{idxCb}][\text{idxCr}][j] = (j == 1) ? 1024 : 0 \\ \quad \text{predCr}[\text{idxY}][\text{idxCb}][\text{idxCr}][j] = (j == 2) ? 1024 : 0 \\ \} \text{ else } \{ \\ \quad \text{predY}[\text{idxY}][\text{idxCb}][\text{idxCr}][j] = \text{LutY}[\text{idxY} - 1][\text{idxCb}][\text{idxCr}][j] \\ \quad \text{predCb}[\text{idxY}][\text{idxCb}][\text{idxCr}][j] = \text{LutCb}[\text{idxY} - 1][\text{idxCb}][\text{idxCr}][j] \\ \quad \text{predCr}[\text{idxY}][\text{idxCb}][\text{idxCr}][j] = \text{LutCr}[\text{idxY} - 1][\text{idxCb}][\text{idxCr}][j] \\ \} \end{aligned} \quad (\text{F-51})$$

$$\begin{aligned}
\text{LutY}[\text{idxY}][\text{idxCb}][\text{idxCr}][j] &= \\
&(\text{cmResCoeff}[\text{idxY}][\text{idxCb}][\text{idxCr}][j][0]) + \\
&\text{predY}[\text{idxY}][\text{idxCb}][\text{idxCr}][j] \\
\text{LutCb}[\text{idxY}][\text{idxCb}][\text{idxCr}][j] &= \\
&(\text{cmResCoeff}[\text{idxY}][\text{idxCb}][\text{idxCr}][j][1]) + \\
&\text{predCb}[\text{idxY}][\text{idxCb}][\text{idxCr}][j] \\
\text{LutCr}[\text{idxY}][\text{idxCb}][\text{idxCr}][j] &= \\
&(\text{cmResCoeff}[\text{idxY}][\text{idxCb}][\text{idxCr}][j][2]) + \\
&\text{predCr}[\text{idxY}][\text{idxCb}][\text{idxCr}][j]
\end{aligned}$$

It is a requirement of bitstream conformance that the values of $\text{LutY}[\text{idxY}][\text{idxCb}][\text{idxCr}][i]$, $\text{LutCb}[\text{idxY}][\text{idxCb}][\text{idxCr}][i]$ and $\text{LutCr}[\text{idxY}][\text{idxCb}][\text{idxCr}][i]$ shall be in the range of -2^{11} to $2^{11}-1$, inclusive.

NOTE – When BitDepthCmInputC is not equal to BitDepthCmInputY , the encoder should select values of $\text{LutY}[\text{idxY}][\text{idxCb}][\text{idxCr}][1]$, $\text{LutY}[\text{idxY}][\text{idxCb}][\text{idxCr}][2]$, $\text{LutCb}[\text{idxY}][\text{idxCb}][\text{idxCr}][0]$ and $\text{LutCr}[\text{idxY}][\text{idxCb}][\text{idxCr}][0]$ that compensate for the bit depth difference.

F.7.4.3.4 Supplemental enhancement information RBSP semantics

The specifications in clause 7.4.3.4 apply.

F.7.4.3.5 Access unit delimiter RBSP semantics

The specifications in clause 7.4.3.5 apply.

F.7.4.3.6 End of sequence RBSP semantics

The specifications in clause 7.4.3.6 apply with the following additions:

When included in a NAL unit with nuh_layer_id nuhLayerId , the end of sequence RBSP specifies that the next picture having nuh_layer_id equal to nuhLayerId or $\text{IdPredictedLayer}[\text{nuhLayerId}][i]$ for any value of i in the range of 0 to $\text{NumPredictedLayers}[\text{nuhLayerId}] - 1$, inclusive, following the current access unit in the bitstream in decoding order (if any) is an IRAP picture with NoRasOutputFlag equal to 1 and with nuh_layer_id equal to nuhLayerId .

F.7.4.3.7 End of bitstream RBSP semantics

The specifications in clause 7.4.3.7 apply.

F.7.4.3.8 Filler data RBSP semantics

The specifications in clause 7.4.3.8 apply.

F.7.4.3.9 Slice segment layer RBSP semantics

The specifications in clause 7.4.3.9 apply.

F.7.4.3.10 RBSP slice segment trailing bits semantics

The specifications in clause 7.4.3.10 apply.

F.7.4.3.11 RBSP trailing bits semantics

The specifications in clause 7.4.3.11 apply.

F.7.4.3.12 Byte alignment semantics

The specifications in clause 7.4.3.12 apply.

F.7.4.4 Profile, tier and level semantics

The specifications in clause 7.4.4 apply with the replacement of each reference to "Annex A" with a reference to "Annex A, clause G.11 or clause H.11" and with the following additions and modifications:

The variable offsetVal is set equal to 0 and the variable $\text{VpsProfileTierLevel}[i]$ is derived according to the following ordered steps:

1. $\text{VpsProfileTierLevel}[0]$ is set equal to the $\text{profile_tier_level}()$ syntax structure in the base VPS (i.e., in the VPS but not in the VPS extension syntax structure $\text{vps_extension}()$) and offsetVal is set equal to 1.

2. When `vps_max_layers_minus1` is greater than 0 and `vps_base_layer_internal_flag` is equal to 1 `VpsProfileTierLevel[offsetVal]` is set equal to the first `profile_tier_level()` syntax structure in the VPS extension, and `offsetVal` is incremented by 1.
3. For `j` in the range of 0 to $(\text{vps_num_profile_tier_level_minus1} - (\text{vps_base_layer_internal_flag} ? 2 : 1))$, inclusive, `VpsProfileTierLevel[offsetVal + j]` is set equal to the `j`-th `profile_tier_level()` syntax structure signalled after the syntax element `vps_num_profile_tier_level_minus1` in the VPS extension.

If `vps_base_layer_internal_flag` is equal to 0, all bits in the `profile_tier_level()` syntax structure `VpsProfileTierLevel[0]` shall be equal to 0 and decoders shall ignore the syntax structure `VpsProfileTierLevel[0]`. Otherwise, the semantics of the `profile_tier_level()` syntax structure `VpsProfileTierLevel[0]` are specified by the remaining part of the current clause.

The scope to which the `profile_tier_level()` syntax structure applies is specified as follows:

- If the `profile_tier_level()` syntax structure is included in an active SPS for the base layer or is the `profile_tier_level()` syntax structure `VpsProfileTierLevel[0]`, it applies to the OLS containing all layers in the bitstream but with only the base layer being the output layer.
- Otherwise, if the `profile_tier_level()` syntax structure is included in an active SPS for an independent non-base layer with `nuh_layer_id` equal to `layerId`, it applies to the output bitstream of the independent non-base layer rewriting process of clause F.10.2 with the input variables `assignedBaseLayerId` equal to `layerId` and `tIdTarget` equal to 6.
- Otherwise, if all of the following conditions are true, the `profile_tier_level()` syntax structure `VpsProfileTierLevel[profile_tier_level_idx[olsIdx][0]]` applies to the output bitstream of the independent non-base layer rewriting process of clause F.10.2 with the input variables `assignedBaseLayerId` equal to `LayerSetLayerIdList[OlsIdxToLsIdx[olsIdx][0]]` and `tIdTarget` equal to 6:
 - `num_add_layer_sets` is greater than 0.
 - `OlsIdxToLsIdx[olsIdx]` is in the range of `FirstAddLayerSetIdx` to `LastAddLayerSetIdx`, inclusive.
 - `NumLayersInLsList[OlsIdxToLsIdx[olsIdx]]` is equal to 1.
 - The `profile_tier_level()` syntax structure `VpsProfileTierLevel[profile_tier_level_idx[olsIdx][0]]` indicates a profile specified in Annex A.
 - The value of `general_inbld_flag` in the `profile_tier_level()` syntax structure `VpsProfileTierLevel[profile_tier_level_idx[olsIdx][0]]` is equal to 1.
- Otherwise, the `profile_tier_level()` syntax structure provides profile, tier and level to which a layer in an OLS conforms and the `profile_tier_level_idx[i][j]` syntax element specifies that the `profile_tier_level()` syntax structure `VpsProfileTierLevel[profile_tier_level_idx[i][j]]` applies to the `j`-th layer of the `i`-th OLS.

NOTE 1 – When `vps_base_layer_internal_flag` is equal to 1 and `vps_max_layers_minus1` is greater than 0, the value of `profile_tier_level_idx[0][0]` is inferred to be equal to 1, such that `VpsProfileTierLevel[1]` applies to the 0-th OLS, which contains only the base layer that is the only output layer.

When the `profile_tier_level()` syntax structure is `VpsProfileTierLevel[k]` for `k` greater than 0 and `profilePresentFlag` is equal to 0, the syntax elements `general_profile_space`, `general_tier_flag`, `general_profile_idc`, `general_profile_compatibility_flag[j]`, `general_progressive_source_flag`, `general_interlaced_source_flag`, `general_non_packed_constraint_flag`, `general_frame_only_constraint_flag`, `general_max_12bit_constraint_flag`, `general_max_10bit_constraint_flag`, `general_max_8bit_constraint_flag`, `general_max_422chroma_constraint_flag`, `general_max_420chroma_constraint_flag`, `general_max_monochrome_constraint_flag`, `general_intra_constraint_flag`, `general_one_picture_only_constraint_flag`, `general_lower_bit_rate_constraint_flag`, `general_max_14bit_constraint_flag`, `general_reserved_zero_33bits`, `general_reserved_zero_34bits`, `general_reserved_zero_7bits`, `general_reserved_zero_35bits`, `general_reserved_zero_43bits`, `general_inbld_flag` and `general_reserved_zero_1bit` are not present in the `profile_tier_level()` syntax structure and each is inferred to be equal to the value of corresponding syntax element of the `profile_tier_level()` syntax structure `VpsProfileTierLevel[k – 1]`.

When the `profile_tier_level()` syntax structure is `VpsProfileTierLevel[k]` for `k` greater than 0 and any of the syntax elements `sub_layer_profile_space[i]`, `sub_layer_tier_flag[i]`, `sub_layer_profile_idc[i]`, `sub_layer_profile_compatibility_flag[i][j]`, `sub_layer_progressive_source_flag[i]`, `sub_layer_interlaced_source_flag[i]`, `sub_layer_non_packed_constraint_flag[i]`, `sub_layer_frame_only_constraint_flag[i]`, `sub_layer_max_12bit_constraint_flag[i]`, `sub_layer_max_10bit_constraint_flag[i]`, `sub_layer_max_8bit_constraint_flag[i]`, `sub_layer_max_422chroma_constraint_flag[i]`, `sub_layer_max_420chroma_constraint_flag[i]`, `sub_layer_max_monochrome_constraint_flag[i]`, `sub_layer_intra_constraint_flag[i]`, `sub_layer_one_picture_only_constraint_flag[i]`, `sub_layer_lower_bit_rate_constraint_flag[i]`, `sub_layer_max_14bit_constraint_flag[i]`, `sub_layer_reserved_zero_33bits[i]`, `sub_layer_reserved_zero_34bits[i]`, `sub_layer_reserved_zero_7bits[i]`, `sub_layer_reserved_zero_35bits[i]`, `sub_layer_reserved_zero_43bits[i]`, `sub_layer_inbld_flag[i]`, `sub_layer_reserved_zero_1bit[i]` and `sub_layer_level_idc[i]` is not present for any value of `i` in the range of 0 to `maxNumSubLayersMinus1 – 1`, inclusive, in the `profile_tier_level()` syntax structure, the value of the syntax element is inferred to be equal to the value of the corresponding syntax element of the `profile_tier_level()` syntax structure `VpsProfileTierLevel[k – 1]`.

A sequence of pictures picSeq is derived as follows:

- If the profile_tier_level() syntax structure is included in an SPS, picSeq consists of the pictures in the CLVS of the layer for which the SPS is the active SPS.
- Otherwise, if the profile_tier_level() syntax structure is VpsProfileTierLevel[0], picSeq consists of the pictures with nuh_layer_id equal to 0 in the CVS.
- Otherwise, if vps_max_layers_minus1 is greater than 0, vps_base_layer_internal_flag is equal to 1 and the profile_tier_level() syntax structure is VpsProfileTierLevel[1], picSeq consists of the pictures with nuh_layer_id equal to 0 in the CVS.
- Otherwise (the profile_tier_level() syntax structure is VpsProfileTierLevel[offsetVal + profile_tier_level_idx[i][j]] for any values of i in the range of 0 to NumOutputLayerSets – 1, inclusive, and j in the range of 0 to NumLayersInIdList[OlsIdxToLsIdx[i]] – 1, inclusive, such that NecessaryLayerFlag[i][j] is equal to 1), picSeq consists of the pictures with nuh_layer_id equal to LayerSetLayerIdList[OlsIdxToLsIdx[i]][j] in the CVS.

general_progressive_source_flag and **general_interlaced_source_flag** are interpreted as follows:

- If general_progressive_source_flag is equal to 1 and general_interlaced_source_flag is equal to 0, the source scan type of the pictures in the picSeq should be interpreted as progressive only.
- Otherwise, if general_progressive_source_flag is equal to 0 and general_interlaced_source_flag is equal to 1, the source scan type of the pictures in the picSeq should be interpreted as interlaced only.
- Otherwise, if general_progressive_source_flag is equal to 0 and general_interlaced_source_flag is equal to 0, the source scan type of the pictures in the picSeq should be interpreted as unknown or unspecified.
- Otherwise, general_progressive_source_flag is equal to 1 and general_interlaced_source_flag is equal to 1, the source scan type of each picture in the picSeq is indicated at the picture level using the syntax element source_scan_type in a picture timing SEI message or the syntax element ffinfo_source_scan_type in a frame-field information SEI message.

NOTE 2 – Decoders may ignore the values of general_progressive_source_flag and general_interlaced_source_flag for purposes other than determining the value to be inferred for frame_field_info_present_flag when vui_parameters_present_flag is equal to 0, as there are no other decoding process requirements associated with the values of these flags. Moreover, the actual source scan type of the pictures is outside the scope of this Specification and the method by which the encoder selects the values of general_progressive_source_flag and general_interlaced_source_flag is unspecified.

general_non_packed_constraint_flag equal to 1 specifies that there are neither frame packing arrangement SEI messages nor segmented rectangular frame packing arrangement SEI messages present for the pictures of picSeq. general_non_packed_constraint_flag equal to 0 indicates that there may or may not be one or more frame packing arrangement SEI messages or segmented rectangular frame packing arrangement SEI messages present for the pictures of picSeq.

NOTE 3 – Decoders may ignore the value of general_non_packed_constraint_flag, as there are no decoding process requirements associated with the presence or interpretation of frame packing arrangement SEI messages or segmented rectangular frame packing arrangement SEI messages.

general_frame_only_constraint_flag equal to 1 specifies that field_seq_flag in the active SPSs for the pictures of picSeq is equal to 0. general_frame_only_constraint_flag equal to 0 indicates that field_seq_flag in the active SPSs for the pictures of picSeq may or may not be equal to 0.

NOTE 4 – Decoders may ignore the value of general_frame_only_constraint_flag, as there are no decoding process requirements associated with the value of field_seq_flag.

NOTE 5 – When general_progressive_source_flag is equal to 1, general_frame_only_constraint_flag may or may not be equal to 1.

It is a requirement of bitstream conformance that when olsIdx is such that alt_output_layer_flag[olsIdx] is equal to 1, general_progressive_source_flag, general_interlaced_source_flag, general_non_packed_constraint_flag and general_frame_only_constraint_flag in VpsProfileTierLevel[offsetVal + profile_tier_level_idx[olsIdx][j]] shall be equal to general_progressive_source_flag, general_interlaced_source_flag, general_non_packed_constraint_flag and general_frame_only_constraint_flag, respectively, in VpsProfileTierLevel[offsetVal + profile_tier_level_idx[olsIdx][k]] for any values of j and k in the range of 0 to NumLayersInIdList[OlsIdxToLsIdx[olsIdx]] – 1, inclusive, such that NecessaryLayerFlag[olsIdx][j] is equal to 1 and NecessaryLayerFlag[olsIdx][k] is equal to 1.

F.7.4.5 Scaling list data semantics

The specifications in clause 7.4.5 apply.

F.7.4.6 Supplemental enhancement information message semantics

The specifications in clause 7.4.6 apply.

F.7.4.7 Slice segment header semantics

F.7.4.7.1 General slice segment header semantics

The specifications in clause 7.4.7.1 apply with the replacement of references to Annex C with references to clause F.13 and with the following additions:

When present, the value of the slice segment header syntax elements `discardable_flag`, `cross_layer_bla_flag`, `inter_layer_pred_enabled_flag`, `num_inter_layer_ref_pics_minus1`, `poc_reset_idc`, `poc_reset_period_id`, `full_poc_reset_flag`, `poc_lsb_val`, `poc_msb_cycle_val_present_flag` and `poc_msb_cycle_val` shall be the same in all slice segment headers of a coded picture. When present, the value of the slice segment header syntax elements `inter_layer_pred_layer_idc[i]` shall be the same in all slice segment headers of a coded picture for each possible value of `i`.

When `nal_unit_type` has a value in the range of `BLA_W_LP` to `RSV_IRAP_VCL23`, inclusive, i.e., the picture is an IRAP picture, `NumDirectRefLayers[nuh_layer_id]` is equal to 0, and `pps_curr_pic_ref_enabled_flag` is equal to 0, `slice_type` shall be equal to 2.

When `sps_max_dec_pic_buffering_minus1[TemporalId]` is equal to 0, `NumDirectRefLayers[nuh_layer_id]` is equal to 0, and `pps_curr_pic_ref_enabled_flag` is equal to 0, `slice_type` shall be equal to 2.

discardable_flag equal to 1 specifies that the coded picture is not used as a reference picture for inter prediction and is not used as a source picture for inter-layer prediction in the decoding process of subsequent pictures in decoding order. `discardable_flag` equal to 0 specifies that the coded picture may be used as a reference picture for inter prediction and may be used as a source picture for inter-layer prediction in the decoding process of subsequent pictures in decoding order. When `nal_unit_type` is equal to `TRAIL_R`, `TSA_R`, `STSA_R`, `RASL_R` or `RADL_R`, the value of `discardable_flag` shall be equal to 0. When not present, the value of `discardable_flag` is inferred to be equal to 0.

NOTE 1 – When a coded picture has `discardable_flag` equal to 1 in its slice headers, the picture may be ignored by decoders without affecting the decoding result of any other picture, in the same layer or a different layer.

cross_layer_bla_flag equal to 1 affects the derivation of `NoClnsOutputFlag` and `LayerResetFlag` as specified in clause F.8.1.3. `cross_layer_bla_flag` shall be equal to 0 for pictures with `nal_unit_type` not equal to `IDR_W_RADL` or `IDR_N_LP` or with `nuh_layer_id` `nuhLayerId` not equal to any value for which `NumDirectRefLayers[nuhLayerId]` is equal to 0. When not present, the value of `cross_layer_bla_flag` is inferred to be equal to 0.

When `vps_extension_flag` is equal to 1, the sum of `NumNegativePics[CurrRpsIdx]`, `NumPositivePics[CurrRpsIdx]`, `num_long_term_sps` and `num_long_term_pics` shall be less than or equal to `MaxDpbSize – 1`, where `MaxDpbSize` is as specified in clause G.11.2 or H.11.2.

NOTE 2 – The way that `MaxDpbSize` is specified in clause G.11.2 is the same as in clause H.11.2.

inter_layer_pred_enabled_flag equal to 1 specifies that inter-layer prediction may be used in decoding of the current picture. `inter_layer_pred_enabled_flag` equal to 0 specifies that inter-layer prediction is not used in decoding of the current picture.

num_inter_layer_ref_pics_minus1 plus 1 specifies the number of pictures that may be used in decoding of the current picture for inter-layer prediction. The length of the `num_inter_layer_ref_pics_minus1` syntax element is $\text{Ceil}(\text{Log2}(\text{NumDirectRefLayers}[\text{nuh_layer_id}]))$ bits. The value of `num_inter_layer_ref_pics_minus1` shall be in the range of 0 to `NumDirectRefLayers[nuh_layer_id] – 1`, inclusive.

The variables `numRefLayerPics` and `refLayerPicIdc[j]` are derived as follows:

```
for( i = 0, j = 0; i < NumDirectRefLayers[ nuh_layer_id ]; i++ ) {  
    refLayerIdx = LayerIdxInVps[ IdDirectRefLayer[ nuh_layer_id ][ i ] ]  
    if( sub_layers_vps_max_minus1[ refLayerIdx ] >= TemporalId && ( TemporalId == 0 || (F-52)  
        max_tid_il_ref_pics_plus1[ refLayerIdx ][ LayerIdxInVps[ nuh_layer_id ] ] >  
        TemporalId ) )  
        refLayerPicIdc[ j++ ] = i  
}  
numRefLayerPics = j
```

The variable `NumActiveRefLayerPics` is derived as follows:

```
if( nuh_layer_id == 0 || numRefLayerPics == 0 )  
    NumActiveRefLayerPics = 0  
else if( default_ref_layers_active_flag )  
    NumActiveRefLayerPics = numRefLayerPics  
else if( !inter_layer_pred_enabled_flag )  
    NumActiveRefLayerPics = 0
```

(F-53)

```

else if( max_one_active_ref_layer_flag || NumDirectRefLayers[ nuh_layer_id ] == 1 )
    NumActiveRefLayerPics = 1
else
    NumActiveRefLayerPics = num_inter_layer_ref_pics_minus1 + 1

```

All slices of a coded picture shall have the same value of NumActiveRefLayerPics.

inter_layer_pred_layer_idc[i] specifies the variable, RefPicLayerId[i], representing the nuh_layer_id of the i-th picture that may be used by the current picture for inter-layer prediction. The length of the inter_layer_pred_layer_idc[i] syntax element is $\text{Ceil}(\text{Log}_2(\text{NumDirectRefLayers}[\text{nuh_layer_id}]))$ bits. The value of inter_layer_pred_layer_idc[i] shall be in the range of 0 to NumDirectRefLayers[nuh_layer_id] - 1, inclusive. When i is greater than 0, inter_layer_pred_layer_idc[i] shall be greater than inter_layer_pred_layer_idc[i - 1]. When not present, the value of inter_layer_pred_layer_idc[i] is inferred to be equal to refLayerPicIdc[i].

The variables RefPicLayerId[i] for all values of i in the range of 0 to NumActiveRefLayerPics - 1, inclusive, are derived as follows:

```

for( i = 0; i < NumActiveRefLayerPics; i++ )                                     (F-54)
    RefPicLayerId[ i ] = IdDirectRefLayer[ nuh_layer_id ][ inter_layer_pred_layer_idc[ i ] ]

```

It is a requirement of bitstream conformance that for each value of i in the range of 0 to NumActiveRefLayerPics - 1, inclusive, either of the following two conditions shall be true:

- The value of max_tid_il_ref_pics_plus1[LayerIdxInVps[RefPicLayerId[i]]][LayerIdxInVps[nuh_layer_id]] is greater than TemporalId.
- The values of max_tid_il_ref_pics_plus1[LayerIdxInVps[RefPicLayerId[i]]][LayerIdxInVps[nuh_layer_id]] and TemporalId are both equal to 0 and the picture in the current access unit with nuh_layer_id equal to RefPicLayerId[i] is an IRAP picture.

poc_reset_idc equal to 0 specifies that neither the most significant bits nor the least significant bits of the picture order count value for the current picture are reset. poc_reset_idc equal to 1 specifies that only the most significant bits of the picture order count value for the current picture may be reset. poc_reset_idc equal to 2 specifies that both the most significant bits and the least significant bits of the picture order count value for the current picture may be reset. poc_reset_idc equal to 3 specifies that either only the most significant bits or both the most significant bits and the least significant bits of the picture order count value for the current picture may be reset and additional picture order count information is signalled. When not present, the value of poc_reset_idc is inferred to be equal to 0.

It is a requirement of bitstream conformance that the following constraints apply:

- The value of poc_reset_idc shall not be equal to 1 or 2 for a RASL, RADL, or SLNR picture or a picture that has TemporalId greater than 0, or a picture that has discardable_flag equal to 1.
- The value of poc_reset_idc of all coded pictures that are present in the bitstream in an access unit shall be the same.
- When the picture in an access unit with nuh_layer_id equal to 0 is an IRAP picture and vps_base_layer_internal_flag is equal to 1 and there is at least one other picture in the same access unit that is not an IRAP picture, the value of poc_reset_idc shall be equal to 1 or 2 for all pictures in the access unit.
- When there is at least one picture that has nuh_layer_id greater than 0 and that is an IDR picture with a particular value of nal_unit_type in an access unit and there is at least one other coded picture that is present in the bitstream in the same access unit with a different value of nal_unit_type, the value of poc_reset_idc shall be equal to 1 or 2 for all pictures in the access unit.
- The value of poc_reset_idc of a CRA or BLA picture shall be less than 3.
- When the picture with nuh_layer_id equal to 0 in an access unit is an IDR picture and vps_base_layer_internal_flag is equal to 1 and there is at least one non-IDR picture in the same access unit, the value of poc_reset_idc shall be equal to 2 for all pictures in the access unit.
- When the picture with nuh_layer_id equal to 0 in an access unit is not an IDR picture and vps_base_layer_internal_flag is equal to 1, the value of poc_reset_idc shall not be equal to 2 for any picture in the access unit.
- When poc_lsb_not_present_flag[LayerIdxInVps[nuh_layer_id]] is equal to 1 and slice_pic_order_cnt_lsb is greater than 0, the value of poc_reset_idc shall not be equal to 2.

The value of poc_reset_idc of an access unit is the value of poc_reset_idc of the pictures in the access unit.

NOTE 3 – Encoders should be cautious in setting the value of poc_reset_idc of pictures when there is a picture of a layer not present in an access unit or there is a picture with discardable_flag equal to 1 present in an access unit, to ensure that the derived picture

order count values of pictures within an access unit are cross-layer aligned. Basically, such cases should be treated similarly as access units with non-cross-layer-aligned IRAP pictures in terms of whether POC resetting is needed.

NOTE 4 – As specified in clause F.8.3.1, the most significant bits of the PicOrderCntVal of an IDR picture with poc_reset_idc equal to 0 are set equal to 0. As specified for the decoder conformance of output order DPB in clause F.13.5.2.2, an IDR picture with nuh_layer_id equal to SmallestLayerId does not cause the output of earlier access units, in decoding order, unless the IDR picture is a POC resetting picture, activates a new VPS, or causes NoClasOutputFlag to be derived to be equal to 1. Encoders should be cautious when using poc_reset_idc equal to 0 with IDR pictures to maintain the desired picture output order and to obey the constraints on the consistent relation of output times to picture order counts as well as the constraints on the output order of two pictures in different CVSs.

poc_reset_period_id identifies a POC resetting period. When not present, the value of poc_reset_period_id is inferred as follows:

- If there is no picture that is in the same layer as the current picture, precedes the current picture in decoding order, and has poc_reset_period_id present in the slice segment header, the value of poc_reset_period_id is inferred to be equal to 0.
- Otherwise, the value of poc_reset_period_id is inferred to be equal to poc_reset_period_id of the last of such pictures, in decoding order, that are in the same layer as the current picture, precede the current picture in decoding order, and has poc_reset_period_id present in the slice segment header.

NOTE 5 – It is not prohibited for multiple pictures in a layer to have the same value of poc_reset_period_id and to have poc_reset_idc equal to 1 or 2 unless such pictures occur in two consecutive access units in decoding order. To minimize the likelihood of such two pictures appearing in the bitstream due to picture losses, bitstream extraction, seeking, or splicing operations, encoders should set the value of poc_reset_period_id to be a random value for each POC resetting period (subject to the constraints specified above).

It is a requirement of bitstream conformance that the following constraints apply:

- There shall be no two pictures consecutive in decoding order in the same layer that have the same value of poc_reset_period_id and poc_reset_idc equal to 1 or 2.
- One POC resetting period shall not include more than one access unit with poc_reset_idc equal to 1 or 2.
- An access unit with poc_reset_idc equal to 1 or 2 shall be the first access unit in a POC resetting period.
- A picture that follows, in decoding order, the first POC resetting picture among all layers of a POC resetting period in decoding order shall not precede, in output order, another picture in any layer that precedes the first POC resetting picture in decoding order.
- The value of poc_reset_period_id shall be the same for all pictures in an access unit.

full_poc_reset_flag equal to 1 specifies that both the most significant bits and the least significant bits of the picture order count value for the current picture are reset when the previous picture in decoding order in the same layer does not belong to the same POC resetting period. full_poc_reset_flag equal to 0 specifies that only the most significant bits of the picture order count value for the current picture are reset when the previous picture in decoding order in the same layer does not belong to the same POC resetting period.

It is a requirement of bitstream conformance that when the value of poc_reset_idc of pictures in the first access unit in the same POC resetting period is equal to 1 or 2, the value of full_poc_reset_flag, when present, shall be equal to poc_reset_idc – 1.

poc_lsb_val specifies a value that may be used to derive the picture order count of the current picture. The length of the poc_lsb_val syntax element is log2_max_pic_order_cnt_lsb_minus4 + 4 bits.

When poc_lsb_not_present_flag[LayerIdxInVps[nuh_layer_id]] is equal to 1 and full_poc_reset_flag is equal to 1, the value of poc_lsb_val shall be equal to 0.

It is a requirement of bitstream conformance that, when poc_reset_idc is equal to 3, and the previous picture picA in decoding order that is in the same layer as the current picture, that has poc_reset_idc equal to 1 or 2, and that belongs to the same POC resetting period is present in the bitstream, picA shall be the same picture as the previous picture in decoding order that is in the same layer as the current picture, that is not a RASL, RADL, or SLNR picture, and that has TemporalId equal to 0 and discardable_flag equal to 0, and the value of poc_lsb_val of the current picture shall be equal to the value of slice_pic_order_cnt_lsb of picA.

The variable PocMsbValRequiredFlag is derived as follows:

$$\text{PocMsbValRequiredFlag} = \text{CraOrBlaPicFlag} \ \&\& \ (!\text{vps_poc_lsb_aligned_flag} \ || \ (\text{vps_poc_lsb_aligned_flag} \ \&\& \ \text{NumDirectRefLayers[nuh_layer_id]} == 0)) \quad (\text{F-55})$$

poc_msb_cycle_val_present_flag equal to 1 specifies that **poc_msb_cycle_val** is present. **poc_msb_cycle_val_present_flag** equal to 0 specifies that **poc_msb_cycle_val** is not present. When not present, the value of **poc_msb_cycle_val_present_flag** is inferred as follows:

- If **slice_segment_header_extension_length** is equal to 0, the value of **poc_msb_cycle_val_present_flag** is inferred to be equal to 0.
- Otherwise, if **PocMsbValRequiredFlag** is equal to 1, the value of **poc_msb_cycle_val_present_flag** is inferred to be equal to 1.
- Otherwise, the value of **poc_msb_cycle_val_present_flag** is inferred to be equal to 0.

NOTE 6 – When the current picture is an IDR picture with **nuh_layer_id** equal to 0 and is a POC resetting picture, **vps_poc_lsb_aligned_flag** is equal to 1 and **NumPredictedLayers[0]** is greater than 0, **poc_msb_cycle_val_present_flag** should be equal to 1.

poc_msb_cycle_val specifies the value that may be used to derive the picture order count value of the current picture or used to derive the value used to decrement the picture order count values of previously decoded pictures in the same layer as the current picture. When **vps_poc_lsb_aligned_flag** is equal to 1, **poc_msb_cycle_val** may also specify the value that is used to derive the value used to decrement the picture order count values of previously decoded pictures of the predicted layers of the current layer. The value of **poc_msb_cycle_val** shall be in the range of 0 to $2^{32 - \log_2 \text{max_pic_order_cnt_lsb_minus4} - 4}$, inclusive.

NOTE 7 – For a picture **picA** in layer **layerA**, let **msbAnchorPic** of the picture **picA** be the previous POC resetting picture in the layer **layerA** or the previous IDR picture that belongs to the layer **layerA** and has **poc_msb_cycle_val_present_flag** equal to 0, whichever is closer, in decoding order, to the picture **picA**. The value of **poc_msb_cycle_val** is set as follows:

- If **vps_poc_lsb_aligned_flag** is equal to 0, the following applies:
 - If either of the following conditions is true, the value of **poc_msb_cycle_val** can be any value in the allowed range:
 - **msbAnchorPic** of the current picture is not present.
 - the current picture is the first picture in the current layer that belongs to or follows an access unit that contains an IRAP picture with **nuh_layer_id** equal to **SmallestLayerId** and **NoClasOutputFlag** equal to 1.
 - Otherwise, if the current picture is a POC resetting picture, the value of **poc_msb_cycle_val** is equal to the difference between the values of the most significant bits of the picture order counts of the current picture, as if there was no POC reset operation for the current picture, and **msbAnchorPic** of the current picture.
 - Otherwise (the current picture is not a POC resetting picture), the value of **poc_msb_cycle_val** is equal to the difference between the values of the most significant bits of the picture order count values of the current picture and **msbAnchorPic** of the current picture.
- Otherwise (**vps_poc_lsb_aligned_flag** is equal to 1), the following applies:
 - If the current picture is an IDR picture with **nuh_layer_id** equal to 0 and is a POC resetting picture, encoders should set the value of **poc_msb_cycle_val** as follows:
 - If picture **picB** other than the current picture is present in the current access unit, the difference between the most significant bits of the picture order counts of **picB** and **msbAnchorPic** of **picB**.
 - Otherwise (no picture **picB** other than the current picture is present in the current access unit), the difference between the most significant bits of the picture order counts of **picB** as if it would be present in the current access unit and **msbAnchorPic** of such a **picB**.
 - Otherwise, if the current picture is the first picture in the POC resetting period and has **nuh_layer_id** equal to **SmallestLayerId**, the value of **poc_msb_cycle_val** is equal to the difference between the values of the most significant bits of the picture order counts of the current picture, as if there was no POC reset operation for the current picture, and **msbAnchorPic** of the current picture.
 - Otherwise when the current picture is not a POC resetting picture, the value of **poc_msb_cycle_val** is equal to the difference between the values of the most significant bits of the picture order count values of the current picture and **msbAnchorPic** of the current picture.

slice_segment_header_extension_data_bit may have any value. Decoders shall ignore the value of **slice_segment_header_extension_data_bit**. Its value does not affect the decoding process specified in this version of this Specification.

7.7.4.7.2 Reference picture list modification semantics

The specifications in clause 7.4.7.2 apply with following modifications:

- Equation 7-57 specifying the derivation of **NumPicTotalCurr** is replaced by:

```

NumPicTotalCurr = 0
if( nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP ) {
    for( i = 0; i < NumNegativePics[ CurrRpsIdx ]; i++ )
        if( UsedByCurrPicS0[ CurrRpsIdx ][ i ] )
            NumPicTotalCurr++
    for( i = 0; i < NumPositivePics[ CurrRpsIdx ]; i++ )
        if( UsedByCurrPicS1[ CurrRpsIdx ][ i ] )
            NumPicTotalCurr++
    for( i = 0; i < num_long_term_sps + num_long_term_pics; i++ )
        if( UsedByCurrPicLt[ i ] )
            NumPicTotalCurr++
}
if( pps_curr_pic_ref_enabled_flag )
    NumPicTotalCurr++
NumPicTotalCurr += NumActiveRefLayerPics

```

(F-56)

F.7.4.7.3 Weighted prediction parameters semantics

The specifications in clause 7.4.7.3 apply.

F.7.4.8 Short-term reference picture set semantics

The specifications in clause 7.4.8 apply, with the following additions:

When `vps_extension_flag` is equal to 1, the value of `num_negative_pics` shall be in the range of 0 to `MaxDpbSize - 1`, inclusive, where `MaxDpbSize` is as specified in clause G.11.2 or H.11.2.

When `vps_extension_flag` is equal to 1, the value of `num_positive_pics` shall be in the range of 0 to `MaxDpbSize - 1 - num_negative_pics`, inclusive, where `MaxDpbSize` is as specified in clause G.11.2 or H.11.2.

F.7.4.9 Slice segment data semantics

F.7.4.9.1 General slice segment data semantics

The specifications in clause 7.4.9.1 apply.

F.7.4.9.2 Coding tree unit semantics

The specifications in clause 7.4.9.2 apply.

F.7.4.9.3 Sample adaptive offset semantics

The specifications in clause 7.4.9.3 apply.

F.7.4.9.4 Coding quadtree semantics

The specifications in clause 7.4.9.4 apply.

F.7.4.9.5 Coding unit semantics

The specifications in clause 7.4.9.5 apply.

F.7.4.9.6 Prediction unit semantics

The specifications in clause 7.4.9.6 apply.

F.7.4.9.7 PCM sample semantics

The specifications in clause 7.4.9.7 apply.

F.7.4.9.8 Transform tree semantics

The specifications in clause 7.4.9.8 apply.

F.7.4.9.9 Motion vector difference semantics

The specifications in clause 7.4.9.9 apply.

F.7.4.9.10 Transform unit semantics

The specifications in clause 7.4.9.10 apply.

F.7.4.9.11 Residual coding semantics

The specifications in clause 7.4.9.11 apply.

F.7.4.9.12 Cross-component prediction syntax

The specifications in clause 7.4.9.12 apply.

F.7.4.9.13 Palette mode semantics

The specifications in clause 7.4.9.13 apply.

F.7.4.9.14 Delta QP semantics

The specifications in clause 7.4.9.14 apply.

F.7.4.9.15 Chroma QP offset semantics

The specifications in clause 7.4.9.15 apply.

F.8 Decoding process

F.8.1 General decoding process

F.8.1.1 General

Input to this process is a bitstream. Output of this process is a list of decoded pictures.

The decoding process is specified such that all decoders capable of decoding an OLS where each of the output layers and the reference layers of the output layers is indicated to conform to a specified profile, tier and level will produce numerically identical cropped decoded output pictures when invoking the decoding process associated with those profiles for the OLS. Any decoding process that produces identical cropped decoded output pictures to those produced by the process described herein (with the correct output order or output timing, as specified) conforms to the decoding process requirements of this Specification.

The following applies at the beginning of decoding a CVSG, after activating the VPS RBSP that is active for the entire CVSG and before decoding any VCL NAL units of the CVSG:

- If `vps_extension()` is not present in the active VPS or a decoding process specified in this annex is not in use, clause 8.1.2 is invoked with the CVSG as input.
- Otherwise (`vps_extension()` is present in the active VPS and a decoding process specified in this annex is in use), clause F.8.1.2 is invoked with the CVSG as input.

F.8.1.2 CVSG decoding process

Input to this process is a CVSG. Output of this process is a list of decoded pictures.

The variable `TargetOlsIdx`, which specifies the index to the list of the OLSs specified by the VPS, of the target OLS, is specified as follows:

- If some external means, not specified in this Specification, is available to set `TargetOlsIdx`, `TargetOlsIdx` is set by the external means.
- Otherwise, if the decoding process is invoked in a bitstream conformance test as specified in clause F.13.1, `TargetOlsIdx` is set as specified in clause F.13.1.
- Otherwise, `TargetOlsIdx` is set equal to 0.

The variable `TargetDecLayerSetIdx`, the layer identifier list `TargetOptLayerIdList`, which specifies the list of `nuh_layer_id` values, in increasing order of `nuh_layer_id` values, of the pictures to be output and the layer identifier list `TargetDecLayerIdList`, which specifies the list of `nuh_layer_id` values, in increasing order of `nuh_layer_id` values, of the NAL units to be decoded, are specified as follows:

```
TargetDecLayerSetIdx = OlsIdxToLsIdx[ TargetOlsIdx ]
lsIdx = TargetDecLayerSetIdx
for( i = 0, j = 0, k = 0; i < NumLayersInIdList[ lsIdx ]; i++ ) {
```



```

    if( NecessaryLayerFlag[ TargetOlsIdx ][ i ] )
        TargetDecLayerIdList[ j++ ] = LayerSetLayerIdList[ lsIdx ][ i ]
    if( OutputLayerFlag[ TargetOlsIdx ][ i ] )
        TargetOptLayerIdList[ k++ ] = LayerSetLayerIdList[ lsIdx ][ i ]
}

```

(F-57)

When TargetOlsIdx is set by external means, it is a requirement for the external means that TargetOlsIdx is constrained as follows:

- When vps_base_layer_available_flag is equal to 0, OlsIdxToLsIdx[TargetOlsIdx] shall be in the range of FirstAddLayerSetIdx to LastAddLayerSetIdx, inclusive.
- When vps_base_layer_internal_flag is equal to 0, TargetOlsIdx shall be greater than 0.

The variable HighestTid, which identifies the highest temporal sub-layer to be decoded, is specified as follows:

- If some external means, not specified in this Specification, is available to set HighestTid, HighestTid is set by the external means.
- Otherwise, if the decoding process is invoked in a bitstream conformance test as specified in clause F.13.1 HighestTid is set as specified in clause F.13.1.
- Otherwise, HighestTid is set equal to vps_max_sub_layers_minus1.

The variable SubPicHrdPreferredFlag is either specified by external means, or when not specified by external means, set equal to 0.

The variable SubPicHrdFlag is specified as follows:

- If the decoding process is invoked in a bitstream conformance test as specified in clause F.13.1, SubPicHrdFlag is set as specified in clause F.13.1.
- Otherwise, SubPicHrdFlag is set equal to (SubPicHrdPreferredFlag && sub_pic_hrd_params_present_flag), where sub_pic_hrd_params_present_flag is found in any hrd_parameters() syntax structure that applies to at least one bitstream partition of the output layer set identified by TargetOlsIdx.

A bitstream to be decoded, BitstreamToDecode, is specified as follows:

- If TargetDecLayerSetIdx is less than or equal to vps_num_layer_sets_minus1 and vps_base_layer_internal_flag is equal to 1, the following applies:
 - The sub-bitstream extraction process as specified in clause 10 is applied with the CVSG, HighestTid and TargetDecLayerIdList as inputs, and the output is assigned to a bitstream referred to as BitstreamToDecode.
 - The variable SmallestLayerId is set equal to 0.
- Otherwise, if TargetDecLayerSetIdx is less than or equal to vps_num_layer_sets_minus1 and vps_base_layer_internal_flag is equal to 0, the following applies:
 - The sub-bitstream extraction process as specified in clause F.10.1 is applied with the CVSG, HighestTid and TargetDecLayerIdList as inputs, and the output is assigned to a bitstream referred to as BitstreamToDecode.
 - The variable SmallestLayerId is set equal to 0.
- Otherwise, if TargetDecLayerSetIdx is greater than vps_num_layer_sets_minus1 and NumLayersInIdList[TargetDecLayerSetIdx] is equal to 1, the following applies:
 - The independent non-base layer rewriting process of clause F.10.2 is applied with the CVSG, HighestTid and TargetDecLayerIdList[0] as inputs, and the output is assigned to a bitstream referred to as BitstreamToDecode.
 - The variable SmallestLayerId is set equal to 0.
- Otherwise, the following applies:
 - The sub-bitstream extraction process as specified in clause F.10.3 is applied with the CVSG, HighestTid and TargetDecLayerIdList as inputs, and the output is assigned to a bitstream referred to as BitstreamToDecode.
 - The variable SmallestLayerId is set equal to TargetDecLayerIdList[0].

When vps_base_layer_internal_flag is equal to 0, vps_base_layer_available_flag is equal to 1 and TargetDecLayerSetIdx is in the range of 0 to vps_num_layer_sets_minus1, inclusive, the following applies:

- The size of the sub-DPB for the layer with nuh_layer_id equal to 0 is set equal to 1.

- The values of `pic_width_in_luma_samples`, `pic_height_in_luma_samples`, `chroma_format_idc`, `separate_colour_plane_flag`, `bit_depth_luma_minus8`, `bit_depth_chroma_minus8`, `conf_win_left_offset`, `conf_win_right_offset`, `conf_win_top_offset` and `conf_win_bottom_offset` for decoded pictures with `nuh_layer_id` equal to 0 are set equal to the values of `pic_width_vps_in_luma_samples`, `pic_height_vps_in_luma_samples`, `chroma_format_vps_idc`, `separate_colour_plane_vps_flag`, `bit_depth_vps_luma_minus8`, `bit_depth_vps_chroma_minus8`, `conf_win_vps_left_offset`, `conf_win_vps_right_offset`, `conf_win_vps_top_offset` and `conf_win_vps_bottom_offset` respectively, of the `vps_rep_format_idx[0]`-th `rep_format()` syntax structure in the active VPS.
- The variable `BaseLayerOutputFlag` is set equal to $(\text{TargetOptLayerIdList}[0] == 0)$.
NOTE – The `BaseLayerOutputFlag` is to be sent by an external means to the base layer decoder for controlling the output of base layer decoded pictures. `BaseLayerOutputFlag` equal to 1 indicates that the base layer is an output layer. `BaseLayerOutputFlag` equal to 0 indicates that the base layer is not an output layer.
- The variable `LayerInitializedFlag[i]` is set equal to 0 for all values of `i` from 0 to `vps_max_layer_id`, inclusive, and the variable `FirstPicInLayerDecodedFlag[i]` is set equal to 0 for all values of `i` from 0 to `vps_max_layer_id`, inclusive.

If `TargetOlsIdx` is equal to 0, clause 8.1.3 is repeatedly invoked for each coded picture in `BitstreamToDecode` in decoding order and the decoding process for the current picture takes as inputs the syntax elements and upper-case variables from clause 7.

Otherwise (`TargetOlsIdx` is greater than 0), clause F.8.1.3 is repeatedly invoked for each coded picture in `BitstreamToDecode` in decoding order.

F.8.1.3 Common decoding process for a coded picture

The decoding processes specified in the remainder of this clause apply to each coded picture, referred to as the current picture and denoted by the variable `CurrPic`, in `BitstreamToDecode`.

Depending on the value of `chroma_format_idc`, the number of sample arrays of the current picture is as follows:

- If `chroma_format_idc` is equal to 0, the current picture consists of 1 sample array `SL`.
- Otherwise (`chroma_format_idc` is not equal to 0), the current picture consists of 3 sample arrays `SL`, `SCb`, `SCr`.

When interpreting the semantics of each syntax element in each NAL unit, the term "the bitstream" (or part thereof, e.g., a CVS of the bitstream) refers to `BitstreamToDecode` (or part thereof).

When `vps_base_layer_internal_flag` is equal to 0, `vps_base_layer_available_flag` is equal to 1, `TargetDecLayerSetIdx` is in the range of 0 to `vps_num_layer_sets_minus1`, inclusive, `TemporalId` is less than or equal to `sub_layers_vps_max_minus1[0]`, and the current picture is the first coded picture of an access unit, clause F.8.1.8 is invoked prior to decoding the current picture.

When the current picture is an IRAP picture, the variable `HandleCraAsBlaFlag` is derived as specified in the following:

- If some external means not specified in this Specification is available to set the variable `HandleCraAsBlaFlag` to a value for the current picture, the variable `HandleCraAsBlaFlag` is set equal to the value provided by the external means.
- Otherwise, the variable `HandleCraAsBlaFlag` is set equal to 0.

When the current picture is an IRAP picture and has `nuh_layer_id` equal to `SmallestLayerId`, the following applies:

- The variable `NoClrasOutputFlag` is specified as follows:
 - If the current picture is the first picture in the bitstream, `NoClrasOutputFlag` is set equal to 1.
 - Otherwise, if the current picture is included in the first access unit that follows an access unit including an end of sequence NAL unit with `nuh_layer_id` equal to `SmallestLayerId` or 0 in decoding order, `NoClrasOutputFlag` is set equal to 1.
 - Otherwise, if the current picture is a BLA picture or a CRA picture with `HandleCraAsBlaFlag` equal to 1, `NoClrasOutputFlag` is set equal to 1.
 - Otherwise, if the current picture is an IDR picture with `cross_layer_bla_flag` is equal to 1, `NoClrasOutputFlag` is set equal to 1.
 - Otherwise, if some external means, not specified in this Specification, is available to set `NoClrasOutputFlag`, `NoClrasOutputFlag` is set by the external means.
 - Otherwise, `NoClrasOutputFlag` is set equal to 0.

- When NoCllasOutputFlag is equal to 1, the variable LayerInitializedFlag[i] is set equal to 0 for all values of i from 0 to vps_max_layer_id, inclusive, and the variable FirstPicInLayerDecodedFlag[i] is set equal to 0 for all values of i from 0 to vps_max_layer_id, inclusive.

The variables LayerResetFlag and dolLayerId are derived as follows:

- If the current picture is an IRAP picture and has nuh_layer_id nuhLayerId greater than SmallestLayerId, the following applies:
 - If the current picture is the first picture, in decoding order, that follows an end of sequence NAL unit with nuh_layer_id equal to nuhLayerId, LayerResetFlag is set equal to 1 and dolLayerId is set equal to the nuh_layer_id value of the current NAL unit.
 - Otherwise, if the current picture is a CRA picture with HandleCraAsBlaFlag equal to 1, an IDR picture with cross_layer_bla_flag is equal to 1 or a BLA picture, LayerResetFlag is set equal to 1 and dolLayerId is set equal to the nuh_layer_id value of the current NAL unit.
 - Otherwise, LayerResetFlag is set equal to 0.

NOTE 1 – An end of sequence NAL unit, a CRA picture with HandleCraAsBlaFlag equal to 1, an IDR picture with cross_layer_bla_flag equal to 1, or a BLA picture, each with nuh_layer_id nuhLayerId greater than SmallestLayerId, may be present to indicate a discontinuity of the layer with nuh_layer_id equal to nuhLayerId and its predicted layers.

- When LayerResetFlag is equal to 1, the following applies:
 - The values of LayerInitializedFlag and FirstPicInLayerDecodedFlag are updated as follows:

```

for( i = 0; i < NumPredictedLayers[ dolLayerId ]; i++ ) {
    iLayerId = IdPredictedLayer[ dolLayerId ][ i ]
    LayerInitializedFlag[ iLayerId ] = 0
    FirstPicInLayerDecodedFlag[ iLayerId ] = 0
}

```

(F-58)

- Each picture that is in the DPB and has nuh_layer_id equal to dolLayerId is marked as "unused for reference".
- When NumPredictedLayers[dolLayerId] is greater than 0, each picture that is in the DPB and has nuh_layer_id equal to any value of IdPredictedLayer[dolLayerId][i] for the values of i in the range of 0 to NumPredictedLayers[dolLayerId] – 1, inclusive, is marked as "unused for reference".
- Otherwise, LayerResetFlag is set equal to 0.

When the current picture is an IRAP picture, the following applies:

- If the current picture with a particular value of nuh_layer_id is an IDR picture, a BLA picture, the first picture with that particular value of nuh_layer_id in the bitstream in decoding order or the first picture with that particular value of nuh_layer_id that follows an end of sequence NAL unit with that particular value of nuh_layer_id in decoding order, the variable NoRaslOutputFlag is set equal to 1.
- Otherwise, if LayerInitializedFlag[nuh_layer_id] is equal to 0 and LayerInitializedFlag[refLayerId] is equal to 1 for all values of refLayerId equal to IdDirectRefLayer[nuh_layer_id][j], where j is in the range of 0 to NumDirectRefLayers[nuh_layer_id] – 1, inclusive, the variable NoRaslOutputFlag is set equal to 1.
- Otherwise, the variable NoRaslOutputFlag is set equal to HandleCraAsBlaFlag.

When the current picture is an IRAP picture with NoRaslOutputFlag equal to 1 and one of the following conditions is true, LayerInitializedFlag[nuh_layer_id] is set equal to 1:

- nuh_layer_id is equal to 0.
- LayerInitializedFlag[nuh_layer_id] is equal to 0 and NumDirectRefLayers[nuh_layer_id] is equal to 0.
- LayerInitializedFlag[nuh_layer_id] is equal to 0 and LayerInitializedFlag[refLayerId] is equal to 1 for all values of refLayerId equal to IdDirectRefLayer[nuh_layer_id][j], where j is in the range of 0 to NumDirectRefLayers[nuh_layer_id] – 1, inclusive.

Depending on the value of separate_colour_plane_flag, the decoding process is structured as follows:

- If separate_colour_plane_flag is equal to 0, the following decoding process is invoked a single time with the current picture being the output.
- Otherwise (separate_colour_plane_flag is equal to 1), the following decoding process is invoked three times. Inputs to the decoding process are all NAL units of the coded picture with identical value of colour_plane_id. The decoding

process of NAL units with a particular value of `colour_plane_id` is specified as if only a CVS with monochrome colour format with that particular value of `colour_plane_id` would be present in the bitstream. The output of each of the three decoding processes is assigned to one of the 3 sample arrays of the current picture, with the NAL units with `colour_plane_id` equal to 0, 1 and 2 being assigned to S_L , S_{Cb} and S_{Cr} , respectively.

NOTE 2 – The variable `ChromaArrayType` is derived as equal to 0 when `separate_colour_plane_flag` is equal to 1 and `chroma_format_idc` is equal to 3. In the decoding process, the value of this variable is evaluated resulting in operations identical to that of monochrome pictures (when `chroma_format_idc` is equal to 0).

The following applies for the decoding of the current picture:

- If the current picture has `nuh_layer_id` equal to 0, the following applies:
 - The decoding process for the current picture takes as inputs the syntax elements and upper-case variables from clause F.7.
 - The variables `NumActiveRefLayerPics`, `NumActiveRefLayerPics0` and `NumActiveRefLayerPics1` are set equal to 0.
 - The decoding process for a coded picture with `nuh_layer_id` equal to 0 as specified in clause F.8.1.4 is invoked.
- Otherwise, the following applies:
 - When `vps_base_layer_internal_flag` is equal to 0, `vps_base_layer_available_flag` is equal to 1, `TargetDecLayerSetIdx` is in the range of 0 to `vps_num_layer_sets_minus1`, inclusive, `TemporalId` is less than or equal to `sub_layers_vps_max_minus1[0]`, the current picture is the first coded picture of an access unit and a decoded picture with `nuh_layer_id` equal to 0 is provided by external means for the current access unit, clause F.8.1.9 is invoked after the decoding of the slice segment header of the first slice segment, in decoding order, of the current picture, but prior to decoding any slice segment of the first coded picture of the access unit.
 - For the decoding of the slice segment header of the first slice segment, in decoding order, of the current picture, the decoding process for starting the decoding of a coded picture with `nuh_layer_id` greater than 0 specified in clause F.8.1.5 is invoked.
 - Let `lIdx` be equal to such value for which the `nuh_layer_id` value of the current picture is equal to `LayerSetLayerIdList[OlsIdxToLsIdx[TargetOlsIdx]][lIdx]`.
 - If `general_profile_idc` in the `profile_tier_level()` syntax structure `VpsProfileTierLevel[profile_tier_level_idx[TargetOlsIdx][lIdx]]` is equal to 6, the decoding process for the current picture takes as inputs the syntax elements and upper-case variables from clause G.7 and the decoding process of clause G.8.1.2 is invoked.
 - Otherwise, if `general_profile_idc` in the `profile_tier_level()` syntax structure `VpsProfileTierLevel[profile_tier_level_idx[TargetOlsIdx][lIdx]]` is equal to 7 or 10, the decoding process for the current picture takes as inputs the syntax elements and upper-case variables from clause H.7 and the decoding process of clause H.8.1.2 is invoked.
 - Otherwise, if `general_profile_idc` in the `profile_tier_level()` syntax structure `VpsProfileTierLevel[profile_tier_level_idx[TargetOlsIdx][lIdx]]` is equal to 8, the decoding process for the current picture takes as inputs the syntax elements and upper-case variables from clause I.7 and the decoding process of clause I.8.1.2 is invoked.
 - Otherwise (the current picture belongs to an independent non-base layer), the decoding process for the current picture takes as inputs the syntax elements and upper-case variables from clause F.7 and the decoding process of clause 8.1.3 is invoked by changing the references to clauses 8.2, 8.3, 8.3.1, 8.3.2, 8.3.3, 8.3.4, 8.4, 8.5, 8.6, and 8.7 with clauses F.8.2, F.8.3, F.8.3.1, F.8.3.2, F.8.3.3, F.8.3.4, F.8.4, F.8.5, F.8.6, and F.8.7, respectively.
 - After all slices of the current picture have been decoded, the decoding process for ending the decoding of a coded picture with `nuh_layer_id` greater than 0 specified in clause F.8.1.6 is invoked.

When the current picture is the last coded picture in an access unit in `BitstreamToDecode`, after the decoding of the current picture, prior to the decoding of the next picture, the following applies:

- `PicOutputFlag` is updated as follows:
 - If `alt_output_layer_flag[TargetOlsIdx]` is equal to 1 and the current access unit either does not contain a picture at the output layer or contains a picture at the output layer that has `PicOutputFlag` equal to 0, the following applies:
 - The list `nonOutputLayerPictures` is set to be the list of the pictures of the access unit with `PicOutputFlag` equal to 1 and with `nuh_layer_id` values among the `nuh_layer_id` values of the reference layers of the output layer.

- When the list nonOutputLayerPictures is not empty, the picture with the highest nuh_layer_id value among the list nonOutputLayerPictures is removed from the list nonOutputLayerPictures.
- PicOutputFlag for each picture that is included in the list nonOutputLayerPictures is set equal to 0.
- Otherwise, PicOutputFlag for pictures that are not included in an output layer is set equal to 0.
- When vps_base_layer_internal_flag is equal to 0, vps_base_layer_available_flag is equal to 1 and TargetDecLayerSetIdx is in the range of 0 to vps_num_layer_sets_minus1, inclusive, the following applies:
 - If BaseLayerOutputFlag is equal to 0, the following applies:
 - If alt_output_layer_flag[TargetOlsIdx] is equal to 1, the base layer is a reference layer of the output layer, the access unit does not contain a picture at the output layer or contains a picture at the output layer that has PicOutputFlag equal to 0, and the access unit does not contain a picture at any other reference layer of the output layer, BaseLayerPicOutputFlag is set equal to 1.
 - Otherwise, BaseLayerPicOutputFlag is set equal to 0.
 - Otherwise, BaseLayerPicOutputFlag is set equal to 1.
- NOTE 3 – The BaseLayerPicOutputFlag for each access unit is to be sent by an external means to the base layer decoder for controlling the output of base layer decoded pictures. BaseLayerPicOutputFlag equal to 1 for an access unit specifies that the base layer picture of the access unit is to be output. BaseLayerPicOutputFlag equal to 0 for an access unit specifies that the base layer picture of the access unit is not to be output.
- The sub-DPB for the layer with nuh_layer_id equal to 0 is set to be empty.
- The variable AuOutputFlag that is associated with the current access unit is derived as follows:
 - If at least one picture in the current access unit has PicOutputFlag equal to 1, AuOutputFlag is set equal to 1.
 - Otherwise, AuOutputFlag is set equal to 0.
- The variable PicLatencyCount that is associated with the current access unit is set equal to 0.
- When AuOutputFlag of the current access unit is equal to 1, for each access unit in the DPB that has at least one picture marked as "needed for output" and follows the current access unit in output order, the associated variable PicLatencyCount is set equal to PicLatencyCount + 1.

F.8.1.4 Decoding process for a coded picture with nuh_layer_id equal to 0

The specifications in clause 8.1.3 apply with the following changes:

- Replace the references to clauses 8.2, 8.3, 8.3.1, 8.3.2, 8.3.3, 8.3.4, 8.4, 8.5, 8.6 and 8.7 with clauses F.8.2, F.8.3, F.8.3.1, F.8.3.2, F.8.3.3, F.8.3.4, F.8.4, F.8.5, F.8.6 and F.8.7, respectively.
- At the end of the clause, add item 5 as follows:
 5. When FirstPicInLayerDecodedFlag[0] is equal to 0, FirstPicInLayerDecodedFlag[0] is set equal to 1.

F.8.1.5 Decoding process for starting the decoding of a coded picture with nuh_layer_id greater than 0

Each picture referred to in this clause is a complete coded picture.

The decoding process operates as follows for the current picture CurrPic:

1. The decoding of NAL units is specified in clause F.8.2.
2. The processes in clause F.8.3 specify the following decoding processes using syntax elements in the slice segment layer and above:
 - Variables and functions relating to picture order count are derived in clause F.8.3.1. This needs to be invoked only for the first slice segment of a picture. It is a requirement of bitstream conformance that PicOrderCntVal of each picture in an access unit shall have the same value during and at the end of decoding of the access unit.

NOTE 1 – When the current picture is or succeeds, in decoding order, an IDR picture with nuh_layer_id equal to 0, PicOrderCntVal of a base layer picture picA preceding, in decoding order, the IDR picture with nuh_layer_id equal to 0 may or may not be equal to PicOrderCntVal of the pictures with nuh_layer_id greater than 0 in the access unit containing picA.
 - The decoding process for RPS in clause F.8.3.2 is invoked, wherein only reference pictures with nuh_layer_id equal to that of CurrPic may be marked as "unused for reference" or "used for long-term

reference" and any picture with a different value of `nuh_layer_id` is not marked. This needs to be invoked only for the first slice segment of a picture.

- A picture storage buffer in the DPB is allocated for storage of the decoded sample values of the current picture after the invocation of the in-loop filter process as specified in clause F.8.7. This version of the current decoded picture is referred to as the current decoded picture after the invocation of the in-loop filter process. When `TwoVersionsOfCurrDecPicFlag` is equal to 0 and `pps_curr_pic_ref_enabled_flag` is equal to 1, this picture storage buffer is marked as "used for long-term reference". When `TwoVersionsOfCurrDecPicFlag` is equal to 1, another picture storage buffer in the DPB is allocated for storage of the decoded sample values of the current picture immediately before the invocation of the in-loop filter process as specified in clause F.8.7, and is marked as "used for long-term reference". This version of the current decoded picture is referred to as the current decoded picture before the invocation of the in-loop filter process. This needs to be invoked only for the first slice segment of a picture.

NOTE 2 – When `TwoVersionsOfCurrDecPicFlag` is equal to 0, there is only one version of the current decoded picture. In this case, if `pps_curr_pic_ref_enabled_flag` is equal to 1, the current decoded picture is marked as "used for long-term reference" during the decoding of the current picture and will be marked as "used for short-term reference" at the end of the decoding of the current picture, otherwise it is not marked at all during the decoding of the current picture and will be marked as "used for short-term reference" at the end of the decoding of the current picture. When `TwoVersionsOfCurrDecPicFlag` is equal to 1, there are two versions of the current decoded picture, one of which is marked as "used for long-term reference" during the decoding of the current picture and will be marked as "unused for reference" at the end of the decoding of the current picture, and the other version is not marked at all during the decoding of the current picture and will be marked as "used for short-term reference" at the end of the decoding of the current picture.

- When `FirstPicInLayerDecodedFlag[nuh_layer_id]` is equal to 0, the decoding process for generating unavailable reference pictures for pictures first in decoding order within a layer specified in clause F.8.1.7 is invoked, which needs to be invoked only for the first slice segment of a picture.
- When `FirstPicInLayerDecodedFlag[nuh_layer_id]` is equal to 1 and the current picture is an IRAP picture with `NoRasOutputFlag` equal to 1, the decoding process for generating unavailable reference pictures specified in clause F.8.3.3 is invoked, which needs to be invoked only for the first slice segment of a picture.

F.8.1.6 Decoding process for ending the decoding of a coded picture with `nuh_layer_id` greater than 0

The marking of decoded pictures is modified as specified in the following:

```
for( i = 0; i < NumActiveRefLayerPics0; i++ )
    RefPicSetInterLayer0[ i ] is marked as "used for short-term reference"
for( i = 0; i < NumActiveRefLayerPics1; i++ )
    RefPicSetInterLayer1[ i ] is marked as "used for short-term reference"
```

(F-59)

`PicOutputFlag` is set as follows:

- If `LayerInitializedFlag[nuh_layer_id]` is equal to 0, `PicOutputFlag` is set equal to 0.
- Otherwise, if the current picture is a RASL picture and `NoRasOutputFlag` of the associated IRAP picture is equal to 1, `PicOutputFlag` is set equal to 0.
- Otherwise, `PicOutputFlag` is set equal to `pic_output_flag`.

The current decoded picture after the invocation of the in-loop filter process as specified in clause 8.7 is marked as "used for short-term reference".

When `FirstPicInLayerDecodedFlag[nuh_layer_id]` is equal to 0, `FirstPicInLayerDecodedFlag[nuh_layer_id]` is set equal to 1.

F.8.1.7 Decoding process for generating unavailable reference pictures for pictures first in decoding order within a layer

This process is invoked for a picture with `nuh_layer_id` equal to `layerId`, when `FirstPicInLayerDecodedFlag[layerId]` is equal to 0.

NOTE – The entire specification of the decoding process for CL-RAS pictures is included only for purposes of specifying constraints on the allowed syntax content of such CL-RAS pictures. During the decoding process, any CL-RAS pictures may be ignored, as these pictures are not specified for output and have no effect on the decoding process of any other pictures that are specified for output. However, in the HRD operations as specified in clause F.13, CL-RAS pictures may need to be taken into consideration in the derivation of CPB arrival and removal times.

When this process is invoked, the following applies:

- For each $\text{RefPicSetStCurrBefore}[i]$, with i in the range of 0 to $\text{NumPocStCurrBefore} - 1$, inclusive, that is equal to "no reference picture", a picture is generated as specified in clause 8.3.3.2 and the following applies:
 - The value of PicOrderCntVal for the generated picture is set equal to $\text{PocStCurrBefore}[i]$.
 - The value of PicOutputFlag for the generated picture is set equal to 0.
 - The generated picture is marked as "used for short-term reference".
 - $\text{RefPicSetStCurrBefore}[i]$ is set to be the generated reference picture.
 - The value of nuh_layer_id for the generated picture is set equal to nuh_layer_id .
- For each $\text{RefPicSetStCurrAfter}[i]$, with i in the range of 0 to $\text{NumPocStCurrAfter} - 1$, inclusive, that is equal to "no reference picture", a picture is generated as specified in clause 8.3.3.2 and the following applies:
 - The value of PicOrderCntVal for the generated picture is set equal to $\text{PocStCurrAfter}[i]$.
 - The value of PicOutputFlag for the generated picture is set equal to 0.
 - The generated picture is marked as "used for short-term reference".
 - $\text{RefPicSetStCurrAfter}[i]$ is set to be the generated reference picture.
 - The value of nuh_layer_id for the generated picture is set equal to nuh_layer_id .
- For each $\text{RefPicSetStFoll}[i]$, with i in the range of 0 to $\text{NumPocStFoll} - 1$, inclusive, that is equal to "no reference picture", a picture is generated as specified in clause 8.3.3.2 and the following applies:
 - The value of PicOrderCntVal for the generated picture is set equal to $\text{PocStFoll}[i]$.
 - The value of PicOutputFlag for the generated picture is set equal to 0.
 - The generated picture is marked as "used for short-term reference".
 - $\text{RefPicSetStFoll}[i]$ is set to be the generated reference picture.
 - The value of nuh_layer_id for the generated picture is set equal to nuh_layer_id .
- For each $\text{RefPicSetLtCurr}[i]$, with i in the range of 0 to $\text{NumPocLtCurr} - 1$, inclusive, that is equal to "no reference picture", a picture is generated as specified in clause 8.3.3.2 and the following applies:
 - The value of PicOrderCntVal for the generated picture is set equal to $\text{PocLtCurr}[i]$.
 - The value of $\text{slice_pic_order_cnt_lsb}$ for the generated picture is inferred to be equal to $(\text{PocLtCurr}[i] \& (\text{MaxPicOrderCntLsb} - 1))$.
 - The value of PicOutputFlag for the generated picture is set equal to 0.
 - The generated picture is marked as "used for long-term reference".
 - $\text{RefPicSetLtCurr}[i]$ is set to be the generated reference picture.
 - The value of nuh_layer_id for the generated picture is set equal to nuh_layer_id .
- For each $\text{RefPicSetLtFoll}[i]$, with i in the range of 0 to $\text{NumPocLtFoll} - 1$, inclusive, that is equal to "no reference picture", a picture is generated as specified in clause 8.3.3.2 and the following applies:
 - The value of PicOrderCntVal for the generated picture is set equal to $\text{PocLtFoll}[i]$.
 - The value of $\text{slice_pic_order_cnt_lsb}$ for the generated picture is inferred to be equal to $(\text{PocLtFoll}[i] \& (\text{MaxPicOrderCntLsb} - 1))$.
 - The value of PicOutputFlag for the generated picture is set equal to 0.
 - The generated picture is marked as "used for long-term reference".
 - $\text{RefPicSetLtFoll}[i]$ is set to be the generated reference picture.
 - The value of nuh_layer_id for the generated picture is set equal to nuh_layer_id .

F.8.1.8 Initialization process for an external base layer picture

A decoded picture with nuh_layer_id equal to 0 may be provided by external means. When not provided, no picture with nuh_layer_id equal to 0 is used for inter-layer prediction for the current access unit. When provided, the following applies:

- The variable `LayerInitializedFlag[0]` is set equal to 1 and the variable `FirstPicInLayerDecodedFlag[0]` is set equal to 1.
- The following information of the picture with `nuh_layer_id` equal to 0 for the access unit is provided by external means:
 - The decoded sample values (1 sample array S_L if `chroma_format_idc` is equal to 0 or 3 sample arrays S_L , S_{Cb} , and S_{Cr} otherwise)
 - The value of the variable `BllrapPicFlag`, and when `BllrapPicFlag` is equal to 1, the value of `nal_unit_type` of the decoded picture
 - `BllrapPicFlag` equal to 1 specifies that the decoded picture is an IRAP picture. `BllrapPicFlag` equal to 0 specifies that the decoded picture is a non-IRAP picture.
 - The provided value of `nal_unit_type` of the decoded picture shall be equal to `IDR_W_RADL`, `CRA_NUT`, or `BLA_W_LP`.
 - `nal_unit_type` equal to `IDR_W_RADL` specifies that the decoded picture is an IDR picture.
 - `nal_unit_type` equal to `CRA_NUT` specifies that the decoded picture is a CRA picture.
 - `nal_unit_type` equal to `BLA_W_LP` specifies that the decoded picture is a BLA picture.
- When `BllrapPicFlag` of the picture with `nuh_layer_id` equal to 0 is equal to 1, the following applies for the decoded picture with `nuh_layer_id` equal to 0 for the access unit:
 - The variable `NoRaslOutputFlag` is specified as follows:
 - If `nal_unit_type` is `IDR_W_RADL` or `BLA_W_LP`, the variable `NoRaslOutputFlag` is set equal to 1.
 - Otherwise, if the current access unit is the first access unit in the bitstream in decoding order, the variable `NoRaslOutputFlag` is set equal to 1.
 - Otherwise, the variable `NoRaslOutputFlag` is set equal to 0.
 - The variable `NoClrasOutputFlag` is specified as follows:
 - If the current access unit is the first access unit in the bitstream, `NoClrasOutputFlag` is set equal to 1.
 - Otherwise, if `nal_unit_type` is equal to `BLA_W_LP`, `NoClrasOutputFlag` is set equal to 1.
 - Otherwise, if some external means, not specified in this Specification, is available to set `NoClrasOutputFlag`, `NoClrasOutputFlag` is set by the external means.
 - Otherwise, `NoClrasOutputFlag` is set equal to 0.
 - When `NoClrasOutputFlag` is equal to 1, the variable `LayerInitializedFlag[i]` is set equal to 0 for all values of `i` from 1 to `vps_max_layer_id`, inclusive, and the variable `FirstPicInLayerDecodedFlag[i]` is set equal to 0 for all values of `i` from 1 to `vps_max_layer_id`, inclusive.

F.8.1.9 Decoding process for an external base layer picture

The following applies for the decoded picture with `nuh_layer_id` equal to 0 for the access unit:

- `TemporalId` and `PicOrderCntVal` of the decoded picture with `nuh_layer_id` equal to 0 are set equal to the `TemporalId` and `PicOrderCntVal`, respectively, of any picture with `nuh_layer_id` greater than 0 in the access unit.

NOTE – The constraint on the value of `TemporalId` being required to be equal to 0 for IRAP pictures also applies to pictures with `nuh_layer_id` equal to 0 when `vps_base_layer_internal_flag` is equal to 0.
- The decoded picture with `nuh_layer_id` equal to 0 is stored in the sub-DPB for the layer with `nuh_layer_id` equal to 0 and is marked as "used for long-term reference".

F.8.2 NAL unit decoding process

The specifications in clause 8.2 apply.

F.8.3 Slice decoding processes

F.8.3.1 Decoding process for picture order count

Output of this process is `PicOrderCntVal`, the picture order count of the current picture.

Picture order counts are used to identify pictures, for deriving motion parameters in merge mode and motion vector prediction and for decoder conformance checking (see clause F.13.5).

Each coded picture is associated with a picture order count variable, denoted as `PicOrderCntVal`.

When the current picture is the first picture among all layers of a POC resetting period, the variable `PocDecrementInDPBFlag[i]` is set equal to 0 for each value of `i` in the range of 0 to 62, inclusive.

The variable `pocResetFlag` is derived as follows:

- If the current picture is a POC resetting picture, the following applies:
 - If `vps_poc_lsb_aligned_flag` is equal to 0, `pocResetFlag` is set equal to 1.
 - Otherwise, if `PocDecrementInDPBFlag[nuh_layer_id]` is equal to 1, `pocResetFlag` is set equal to 0.
 - Otherwise, `pocResetFlag` is set equal to 1.
- Otherwise, `pocResetFlag` is set equal to 0.

The list `affectedLayerList` is derived as follows:

- If `vps_poc_lsb_aligned_flag` is equal to 0, `affectedLayerList` consists of the `nuh_layer_id` of the current picture.
- Otherwise, `affectedLayerList` consists of the `nuh_layer_id` of the current picture and the `nuh_layer_id` values equal to `IdPredictedLayer[currNuhLayerId][j]` for all values of `j` in the range of 0 to `NumPredictedLayers[currNuhLayerId] – 1`, inclusive, where `currNuhLayerId` is the `nuh_layer_id` value of the current picture.

Depending on `pocResetFlag`, the following applies:

- If `pocResetFlag` is equal to 1, the following applies:
 - When `FirstPicInLayerDecodedFlag[nuh_layer_id]` is equal to 1, the following applies:
 - The variables `pocMsbDelta`, `pocLsbDelta` and `DeltaPocVal` are derived as follows:


```

              if( poc_reset_idc == 3 )
                pocLsbVal = poc_lsb_val
              else
                pocLsbVal = slice_pic_order_cnt_lsb
              if( poc_msb_cycle_val_present_flag )
                pocMsbDelta = poc_msb_cycle_val * MaxPicOrderCntLsb
              else {
                prevPicOrderCntLsb = PrevPicOrderCnt[ nuh_layer_id ] & ( MaxPicOrderCntLsb – 1 )
                prevPicOrderCntMsb = PrevPicOrderCnt[ nuh_layer_id ] – prevPicOrderCntLsb
                pocMsbDelta = GetCurrMsb( pocLsbVal, prevPicOrderCntLsb, prevPicOrderCntMsb,
                                      MaxPicOrderCntLsb )
              }
              if( poc_reset_idc == 2 || ( poc_reset_idc == 3 && full_poc_reset_flag ) )
                pocLsbDelta = pocLsbVal
              else
                pocLsbDelta = 0
              DeltaPocVal = pocMsbDelta + pocLsbDelta
              
```

(F-60)
 - The `PicOrderCntVal` of each picture that has `nuh_layer_id` value `nuhLayerId` for which `PocDecrementInDPBFlag[nuhLayerId]` is equal to 0 and that is equal to any value in `affectedLayerList` is decremented by `DeltaPocVal`.
 - `PocDecrementInDPBFlag[nuhLayerId]` is set equal to 1 for each value of `nuhLayerId` included in `affectedLayerList`.
- The `PicOrderCntVal` of the current picture is derived as follows:


```

      if( poc_reset_idc == 1 )
        PicOrderCntVal = slice_pic_order_cnt_lsb
      else if( poc_reset_idc == 2 )
        PicOrderCntVal = 0
      else {

```

$$\begin{aligned}
 \text{PicOrderCntMsb} &= \text{GetCurrMsb}(\text{slice_pic_order_cnt_lsb}, \text{full_poc_reset_flag} ? 0 : \text{poc_lsb_val}, \\
 &\quad 0, \text{MaxPicOrderCntLsb}) \\
 \text{PicOrderCntVal} &= \text{PicOrderCntMsb} + \text{slice_pic_order_cnt_lsb} \\
 &\}
 \end{aligned}
 \tag{F-61}$$

- Otherwise (pocResettingFlag is equal to 0), the following applies:
 - The PicOrderCntVal of the current picture is derived as follows:

$$\begin{aligned}
 &\text{if}(\text{poc_msb_cycle_val_present_flag}) \\
 &\quad \text{PicOrderCntMsb} = \text{poc_msb_cycle_val} * \text{MaxPicOrderCntLsb} \\
 &\text{else if}(\text{!FirstPicInLayerDecodedFlag}[\text{nuh_layer_id}] \mid \mid \\
 &\quad \text{nal_unit_type} == \text{IDR_N_LP} \mid \mid \text{nal_unit_type} == \text{IDR_W_RADL}) \\
 &\quad \text{PicOrderCntMsb} = 0 \\
 &\text{else} \{ \\
 &\quad \text{prevPicOrderCntLsb} = \text{PrevPicOrderCnt}[\text{nuh_layer_id}] \& (\text{MaxPicOrderCntLsb} - 1) \\
 &\quad \text{prevPicOrderCntMsb} = \text{PrevPicOrderCnt}[\text{nuh_layer_id}] - \text{prevPicOrderCntLsb} \\
 &\quad \text{PicOrderCntMsb} = \text{GetCurrMsb}(\text{slice_pic_order_cnt_lsb}, \text{prevPicOrderCntLsb}, \\
 &\quad \quad \text{prevPicOrderCntMsb}, \text{MaxPicOrderCntLsb}) \\
 &\} \\
 &\text{PicOrderCntVal} = \text{PicOrderCntMsb} + \text{slice_pic_order_cnt_lsb}
 \end{aligned}
 \tag{F-62}$$

The value of PrevPicOrderCnt[Ild] for each of the Ild values included in affectedLayerList is derived as follows:

- If the current picture is not a RASL, RADL or SLNR picture, and the current picture has TemporalId equal to 0 and discardable_flag equal to 0, PrevPicOrderCnt[Ild] is set equal to PicOrderCntVal.
- Otherwise, when poc_reset_idc is equal to 3 and one of the following conditions is true, PrevPicOrderCnt[Ild] is set equal to (full_poc_reset_flag ? 0 : poc_lsb_val):
 - FirstPicInLayerDecodedFlag[nuh_layer_id] is equal to 0.
 - FirstPicInLayerDecodedFlag[nuh_layer_id] is equal to 1 and the current picture is a POC resetting picture.

The value of PicOrderCntVal shall be in the range of -2^{31} to $2^{31} - 1$, inclusive.

It is a requirement of bitstream conformance that the current PicOrderCntVal values of any two pictures in the same layer and the same CVS shall not be the same.

NOTE 1 – PicOrderCntVal, as derived in this clause for the current picture, may be equal to initialPocVal, where initialPocVal is the PicOrderCntVal derived by this clause when an earlier picture, in decoding order, with nuh_layer_id equal to currNuhLayerId in the current CVS, was the current picture. However, the decoding processes applied subsequently have updated the PicOrderCntVal value for that earlier picture so that it no longer is equal to the PicOrderCntVal value derived in this clause for the current picture.

The function PicOrderCnt(picX) is specified as follows:

$$\text{PicOrderCnt}(\text{picX}) = \text{PicOrderCntVal of the picture picX}
 \tag{F-63}$$

The function DiffPicOrderCnt(picA, picB) is specified as follows:

$$\text{DiffPicOrderCnt}(\text{picA}, \text{picB}) = \text{PicOrderCnt}(\text{picA}) - \text{PicOrderCnt}(\text{picB})
 \tag{F-64}$$

The bitstream shall not contain data that result in values of DiffPicOrderCnt(picA, picB) used in the decoding process that are not in the range of -2^{15} to $2^{15} - 1$, inclusive.

NOTE 2 – Let X be the current picture and Y and Z be two other pictures in the same sequence, Y and Z are considered to be in the same output order direction from X when both DiffPicOrderCnt(X, Y) and DiffPicOrderCnt(X, Z) are positive or both are negative.

F.8.3.2 Decoding process for reference picture set

The specifications in clause 8.3.2 apply with the following changes:

- The references to clauses 7.4.7.2, 8.3.1, 8.3.3 and 8.3.4 are replaced with references to clauses F.7.4.7.2, F.8.3.1, F.8.3.3 and F.8.3.4, respectively.
- The following specifications are added:

- When the current picture is an IRAP picture with `nuh_layer_id` equal to `SmallestLayerId`, all reference pictures with any value of `nuh_layer_id` currently in the DPB (if any) are marked as "unused for reference" when at least one of the following conditions is true:
 - The current picture has `NoClasOutputFlag` is equal to 1.
 - The current picture activates a new VPS.
- It is a requirement of bitstream conformance that the RPS is restricted as follows:
 - When the current picture is a CRA picture, there shall be no picture in `RefPicSetStCurrBefore`, `RefPicSetStCurrAfter` or `RefPicSetLtCurr`.
- The constraints specified in clause 8.3.2 on the value of `NumPicTotalCurr` are replaced with the following:
 - It is a requirement of bitstream conformance that the following applies to the value of `NumPicTotalCurr`:
 - If the current picture is a BLA or CRA picture and either `currPicLayerId` is equal to 0 or `NumDirectRefLayers[currPicLayerId]` is equal to 0, the value of `NumPicTotalCurr` shall be equal to `pps_curr_pic_ref_enabled_flag`.
 - Otherwise, when the current picture contains a P or B slice, the value of `NumPicTotalCurr` shall not be equal to 0.
- The constraint, specified in clause 8.3.2, that requires no entry in `RefPicSetStCurrBefore`, `RefPicSetStCurrAfter`, or `RefPicSetLtCurr` when one or more of three conditions are true is replaced with the following:
 - There shall be no entry in `RefPicSetStCurrBefore`, `RefPicSetStCurrAfter`, or `RefPicSetLtCurr` for which one or more of the following are true:
 - The entry is equal to "no reference picture" and `FirstPicInLayerDecodedFlag[currPicLayerId]` is equal to 1.
 - The entry is an SLNR picture and has `TemporalId` equal to that of the current picture.
 - The entry is a picture that has `TemporalId` greater than that of the current picture.

F.8.3.3 Decoding process for generating unavailable reference pictures

The specifications in clause 8.3.3 and its subclauses apply with the replacement of references to Annex C with references to clause F.13.

F.8.3.4 Decoding process for reference picture lists construction

This process is invoked at the beginning of the decoding process for each P or B slice.

Reference pictures are addressed through reference indices as specified in clause 8.5.3.3.2. A reference index is an index into a reference picture list. When decoding a P slice, there is a single reference picture list `RefPicList0`. When decoding a B slice, there is a second independent reference picture list `RefPicList1` in addition to `RefPicList0`.

At the beginning of the decoding process for each slice, the reference picture lists `RefPicList0` and, for B slices, `RefPicList1` are derived as follows:

If `TwoVersionsOfCurrDecPicFlag` is equal to 1, let the variable `currPic` be the current decoded picture before the invocation of the in-loop filter process; otherwise (`TwoVersionsOfCurrDecPicFlag` is equal to 0), let the variable `currPic` be the current decoded picture after the invocation of the in-loop filter process. The variable `NumRpsCurrTempList0` is set equal to `Max(num_ref_idx_l0_active_minus1 + 1, NumPicTotalCurr)` and the list `RefPicListTemp0` is constructed as follows:

```

rIdx = 0
while( rIdx < NumRpsCurrTempList0 ) {
  for( i = 0; i < NumPocStCurrBefore && rIdx < NumRpsCurrTempList0; rIdx++, i++ )
    RefPicListTemp0[ rIdx ] = RefPicSetStCurrBefore[ i ]
  for( i = 0; i < NumActiveRefLayerPics0; rIdx++, i++ )
    RefPicListTemp0[ rIdx ] = RefPicSetInterLayer0[ i ]
  for( i = 0; i < NumPocStCurrAfter && rIdx < NumRpsCurrTempList0; rIdx++, i++ )      (F-65)
    RefPicListTemp0[ rIdx ] = RefPicSetStCurrAfter[ i ]
  for( i = 0; i < NumPocLtCurr && rIdx < NumRpsCurrTempList0; rIdx++, i++ )
    RefPicListTemp0[ rIdx ] = RefPicSetLtCurr[ i ]
  for( i = 0; i < NumActiveRefLayerPics1; rIdx++, i++ )
    RefPicListTemp0[ rIdx ] = RefPicSetInterLayer1[ i ]
  if( pps_curr_pic_ref_enabled_flag )

```

```

        RefPicListTemp0[ rIdx++ ] = currPic
    }

```

The list RefPicList0 is constructed as follows:

```

for( rIdx = 0; rIdx <= num_ref_idx_l0_active_minus1; rIdx++ )
    RefPicList0[ rIdx ] = ref_pic_list_modification_flag_l0 ? RefPicListTemp0[ list_entry_l0[ rIdx ] ] :
                                                                    RefPicListTemp0[ rIdx ]
if( pps_curr_pic_ref_enabled_flag && !ref_pic_list_modification_flag_l0 &&
    NumRpsCurrTempList0 > ( num_ref_idx_l0_active_minus1 + 1 ) )
    RefPicList0[ num_ref_idx_l0_active_minus1 ] = currPic

```

(F-66)

When the slice is a B slice, the variable NumRpsCurrTempList1 is set equal to Max(num_ref_idx_l1_active_minus1 + 1, NumPicTotalCurr) and the list RefPicListTemp1 is constructed as follows:

```

rIdx = 0
while( rIdx < NumRpsCurrTempList1 ) {
    for( i = 0; i < NumPocStCurrAfter && rIdx < NumRpsCurrTempList1; rIdx++, i++ )
        RefPicListTemp1[ rIdx ] = RefPicSetStCurrAfter[ i ]
    for( i = 0; i < NumActiveRefLayerPics1; rIdx++, i++ )
        RefPicListTemp1[ rIdx ] = RefPicSetInterLayer1[ i ]
    for( i = 0; i < NumPocStCurrBefore && rIdx < NumRpsCurrTempList1; rIdx++, i++ )
        RefPicListTemp1[ rIdx ] = RefPicSetStCurrBefore[ i ]
    for( i = 0; i < NumPocLtCurr && rIdx < NumRpsCurrTempList1; rIdx++, i++ )
        RefPicListTemp1[ rIdx ] = RefPicSetLtCurr[ i ]
    for( i = 0; i < NumActiveRefLayerPics0; rIdx++, i++ )
        RefPicListTemp1[ rIdx ] = RefPicSetInterLayer0[ i ]
    if( pps_curr_pic_ref_enabled_flag )
        RefPicListTemp1[ rIdx++ ] = currPic
}

```

(F-67)

When the slice is a B slice, the list RefPicList1 is constructed as follows:

```

for( rIdx = 0; rIdx <= num_ref_idx_l1_active_minus1; rIdx++ )
    RefPicList1[ rIdx ] = ref_pic_list_modification_flag_l1 ? RefPicListTemp1[ list_entry_l1[ rIdx ] ] :
                                                                RefPicListTemp1[ rIdx ]

```

(F-68)

It is a requirement of bitstream conformance that when the current layer is an independent non-base layer, nal_unit_type has a value in the range of BLA_W_LP to RSV_IRAP_VCL23, inclusive (i.e. the picture is an IRAP picture), pps_curr_pic_ref_enabled_flag is equal to 1, and slice_type is not equal to 2, RefPicList0 and RefPicList1 shall not contain entries that refer to a picture other than the current picture.

F.8.4 Decoding process for coding units coded in intra prediction mode

The specifications in clause 8.4 and its subclauses apply.

F.8.5 Decoding process for coding units coded in inter prediction mode

The specifications in clause 8.5 and its subclauses apply.

F.8.6 Scaling, transformation and array construction process prior to deblocking filter process

The specifications in clause 8.6 and its subclauses apply.

F.8.7 In-loop filter process

The specifications in clause 8.7 and its subclauses apply.

F.9 Parsing process

The specifications in clause 9 and its subclauses apply.

F.10 Specification of bitstream subsets

F.10.1 Sub-bitstream extraction process

Inputs to this process are a bitstream `inBitstream`, a target highest TemporalId value `tIdTarget` and a target layer identifier list `layerIdListTarget`.

Output of this process is a sub-bitstream `outBitstream`.

When `vps_base_layer_internal_flag` is equal to 1 and `vps_base_layer_available_flag` is equal to 1, it is a requirement of bitstream conformance for `inBitstream` that any output sub-bitstream `outBitstream` that is the output of the process specified in this clause with `inBitstream`, `tIdTarget` equal to any value in the range of 0 to 6, inclusive, and `layerIdListTarget` equal to the layer identifier list associated with a layer set specified in the active VPS as inputs, and that satisfies both of the following conditions shall be a conforming bitstream:

- The output sub-bitstream contains at least one VCL NAL unit with `nuh_layer_id` equal to each of the `nuh_layer_id` values in `layerIdListTarget`.
- The output sub-bitstream contains at least one VCL NAL unit with TemporalId equal to `tIdTarget`.

NOTE 1 – When `vps_base_layer_internal_flag` is equal to 1 and `vps_base_layer_available_flag` is equal to 1, a conforming bitstream contains one or more coded slice segment NAL units with `nuh_layer_id` equal to 0 and TemporalId equal to 0.

The output sub-bitstream `outBitstream` is derived as follows:

- The bitstream `outBitstream` is set to be identical to the bitstream `inBitstream`.
- When one or more of the following two conditions are true, remove from `outBitstream` all SEI NAL units that have `nuh_layer_id` equal to 0 and that contain a non-scalable-nested buffering period SEI message, a non-scalable-nested picture timing SEI message, or a non-scalable-nested decoding unit information SEI message:
 - `layerIdListTarget` does not include all the values of `nuh_layer_id` in all NAL units in the bitstream.
 - `tIdTarget` is less than the greatest TemporalId in all NAL units in the bitstream.

NOTE 2 – A "smart" bitstream extractor may include appropriate non-scalable-nested buffering picture SEI messages, non-scalable-nested picture timing SEI messages and non-scalable-nested decoding unit information SEI messages in the extracted sub-bitstream, provided that the SEI messages applicable to the sub-bitstream were present as scalable-nested SEI messages in the original bitstream.

- Remove from `outBitstream` all NAL units with TemporalId greater than `tIdTarget` or `nuh_layer_id` not among the values included in `layerIdListTarget`.

F.10.2 Independent non-base layer rewriting process

Inputs to this process are a bitstream `inBitstream`, a target highest TemporalId value `tIdTarget` and a `nuh_layer_id` value `assignedBaseLayerId` of an independent non-base layer of an additional layer set.

Output of this process is a sub-bitstream `outBitstream`.

It is a requirement of bitstream conformance for `inBitstream` that any output sub-bitstream `outBitstream` that is the output of the process specified in this clause shall otherwise be a conforming bitstream except that `outBitstream` does not contain any VPS NAL units, when all of the following conditions apply:

- The output sub-bitstream `outBitstream` contains VCL NAL units with `nuh_layer_id` equal to `assignedBaseLayerId`.
- The value of `tIdTarget` is equal to any value in the range of 0 to 6, inclusive, and `outBitstream` contains at least one VCL NAL unit with TemporalId equal to `tIdTarget`.
- There is an OLS in the active VPS that consists of only the layer with `nuh_layer_id` equal to `assignedBaseLayerId`, the profile of that layer is a profile specified in Annex A and the value of `general_inbld_flag` (when `tIdTarget` is equal to `vps_max_sub_layers_minus1`) or the value of `sub_layer_inbld_flag[tIdTarget]` (when `tIdTarget` is less than `vps_max_sub_layers_minus1`) in the `profile_tier_level()` syntax structure associated with that layer is equal to 1.

The output sub-bitstream `outBitstream` is derived from the bitstream `inBitstream` as follows:

- The bitstream `outBitstream` is set to be identical to the bitstream `inBitstream`.
- NAL units with `nal_unit_type` not equal to `SPS_NUT`, `PPS_NUT`, and `EOB_NUT` and with `nuh_layer_id` not equal to the `assignedBaseLayerId` are removed from `outBitstream`.
- NAL units with `nal_unit_type` equal to `SPS_NUT` or `PPS_NUT` with `nuh_layer_id` not equal to 0 or `assignedBaseLayerId` are removed from `outBitstream`.
- NAL units with `nal_unit_type` equal to `VPS_NUT` are removed from `outBitstream`.

- All NAL units with TemporalId greater than tIdTarget are removed from outBitstream.
- nuh_layer_id is set equal to 0 in each NAL unit of outBitstream.

F.10.3 Sub-bitstream extraction process for additional layer sets

Inputs to this process are a bitstream inBitstream, a target highest TemporalId value tIdTarget and a target layer identifier list layerIdListTarget of an additional layer set.

Output of this process is a sub-bitstream outBitstream.

It is a requirement of bitstream conformance for the input bitstream that the output sub-bitstream of the process specified in this clause shall be a conforming bitstream according to at least one profile in which vps_base_layer_available_flag may be equal to 0, when all of the following conditions apply:

- The output sub-bitstream outBitstream contains VCL NAL units with each nuh_layer_id value in layerIdListTarget.
- The value of tIdTarget is equal to any value in the range of 0 to 6, inclusive, and outBitstream contains at least one VCL NAL unit with TemporalId equal to tIdTarget.
- layerIdListTarget is identical to LayerSetLayerIdList[i] for any value of i in the range of FirstAddLayerSetIdx to LastAddLayerSetIdx, inclusive.

The output sub-bitstream outBitstream is derived as follows:

- The bitstream outBitstream is set to be identical to the bitstream inBitstream.
- NAL units with nal_unit_type not equal to VPS_NUT, SPS_NUT, PPS_NUT, EOS_NUT and EOB_NUT and with nuh_layer_id not equal to any value in layerIdListTarget are removed from outBitstream.
- NAL units with nal_unit_type equal to VPS_NUT, SPS_NUT, PPS_NUT or EOS_NUT with nuh_layer_id not equal to 0 or any value in layerIdListTarget are removed from outBitstream.
- All NAL units with TemporalId greater than tIdTarget are removed from outBitstream.
- vps_base_layer_available_flag in each VPS is set equal to 0.

F.11 Profiles, tiers and levels

F.11.1 Independent non-base layer decoding capability

This clause specifies the independent non-base layer decoding (INBLD) capability, which is associated with the decoding capability of one or more of the profiles specified in Annex A. When expressing the capabilities of a decoder for one or more profiles specified in Annex A, whether the INBLD capability is supported for those profiles should also be expressed.

NOTE 1– The INBLD capability, when supported, indicates the capability of a decoder to decode an independent non-base layer that is indicated in the active VPSs and SPSs to conform to a profile specified in Annex A and is the layer with the smallest nuh_layer_id value in an additional layer set.

When the profile_tier_level() syntax structure is used for indicating of a decoder capability in systems, the INBLD capability may be indicated by setting the general_inbld_flag equal to 1 in the profile_tier_level() syntax structure used to express the profile, tier and level that the decoder conforms to.

general_inbld_flag is set equal to 1 in the profile_tier_level() syntax structures in which a profile specified in Annex A is indicated and which are either specified in the VPS to be applicable for a non-base layer or included in an SPS activated for an independent non-base layer.

Decoders having the INBLD capability and conforming to a specific profile specified in Annex A at a specific level of a specific tier shall be capable of decoding any independent non-base layer or a sub-layer representation with TemporalId equal to i of the independent non-base layer for which all of the following condition applies for each active VPS:

- There is an OLS that consists of the independent non-base layer and for which the associated profile_tier_level() syntax structure ptlStruct is constrained as follows:
 - ptlStruct indicates that the independent non-base layer or the sub-layer representation conforms to a profile specified in Annex A.
 - ptlStruct indicates that the independent non-base layer or the sub-layer representation conforms to a level lower than or equal to the specified level.
 - ptlStruct indicates that the independent non-base layer or the sub-layer representation conforms to a tier lower than or equal to the specified tier.
 - general_inbld_flag or sub_layer_inbld_flag[i] in ptlStruct is equal to 1.

NOTE 2– Let a derived bitstream `outBitstream` be a bitstream which is derived by invoking the independent non-base layer rewriting process specified in clause F.10.2 with a bitstream containing one or more independent non-base layers, `tldTarget` equal to 6 and `assignedBaseLayerId` equal to the smallest `nuh_layer_id` value of additional layer set as inputs. As specified elsewhere in this Specification, it is a requirement of bitstream conformance that `outBitstream` is otherwise a conforming bitstream but does not contain VPSs. Consequently, decoders with INBLD capability may apply the independent non-base layer rewriting process specified in clause F.10.2 to obtain `outBitstream` and then apply a decoding process for a profile specified in Annex A with `outBitstream` as input.

NOTE 3– The following constraints are necessary to ensure that a sub-bitstream derived by invoking the independent non-base layer rewriting process specified in clause F.10.2 conforms to a profile specified in Annex A:

- All active SPSs for the independent non-base layer have `nuh_layer_id` equal to 0 or `MultiLayerExtSpsFlag` equal to 0.
- Each active VPS has `poc_lsb_not_present_flag[i]` equal to 1 for each value of `i` for which `i` is the layer index of the independent non-base layer.

F.11.2 Decoder capabilities

This clause specifies requirements for decoders having the capability of decoding an output operation point with one or more necessary layers.

NOTE – For example, this clause specifies the requirements for a decoder supporting simultaneous decoding of the base layer and an independent non-base layer for which the INBLD capability is needed. Moreover, this clause specifies the requirements for a decoder supporting decoding of layers conforming to profiles specified in Annex G or H.

A decoder that conforms to a list `ptliList` consisting of profile, tier, level and INBLD capability quadruplets (`Pi`, `Ti`, `Li`, `InbldFlagi`) for `i` in the range of 0 to `d – 1`, inclusive, where `d` is a positive integer, shall be capable of decoding any output operation point containing `b` necessary layers, where `b` is less than or equal to `d`, when the following condition applies:

- There exists a reordered list, `orderedPtliList`, of the profile, tier, level and INBLD capability quadruplets (`orderedPj`, `orderedTj`, `orderedLj`, `orderedInbldFlagj`) of the list `ptliList` such that all the following conditions apply for each value of `j` in the range of 0 to `b – 1`, inclusive, where the variable `bLayer[j]` represents the `j`-th necessary layer in the output operation point:
 - The profile to which `bLayer[j]` is indicated to conform is included in `CompatibleProfileList` for the profile `orderedPj` as specified in Table F.3.
 - The tier to which `bLayer[j]` is indicated to conform is lower than or equal to the tier `orderedTj`.
 - The level to which `bLayer[j]` is indicated to conform is lower than or equal to the level `orderedLj`.
 - When the profile to which `bLayer[j]` is indicated to conform is a profile specified in Annex A, the value of `general_inbld_flag` (when `OpTid` of the output operation point is equal to `vps_max_sub_layers_minus1`) or `sub_layer_inbld_flag[OpTid]` (when `OpTid` of the output operation point is less than `vps_max_sub_layers_minus1`) of `bLayer[j]` is less than or equal to `orderedInbldFlagj`.

For each particular format range extensions profile specified in clause A.3.5, the compatible format range extensions profiles are defined as the profiles that are indicated by both of the following conditions being true:

- The value of `general_profile_idc` is equal to 4 or `general_profile_compatibility_flag[4]` is equal to 1.
- The value of each constraint flag listed in Table A.2 is less than or equal to the corresponding value specified in the row of Table A.2 for the particular format range extensions profile.

For each particular scalable format range extensions profile specified in clause H.11.1.2, the compatible scalable format range extensions profiles are defined as the profiles that are indicated by all of the following conditions being true:

- The value of `general_profile_idc` is equal to 10 or `general_profile_compatibility_flag[10]` is equal to 1.
- The value of each constraint flag listed in Table H.4 is less than or equal to the corresponding value specified in the row of Table H.4 for the particular scalable format range extensions profile.

Table F.3 – Specification of CompatibleProfileList

Profile to which the decoder conforms	Profiles that the decoder shall support CompatibleProfileList
Scalable Main	Scalable Main, Main, Main Still Picture
Scalable Main 10	Scalable Main 10, Main, Main Still Picture, Main 10, Scalable Main
Scalable Monochrome	The compatible format range extensions profiles of the Monochrome profile, and the compatible scalable format range extensions profiles of the Scalable Monochrome profile
Scalable Monochrome 12	The compatible format range extensions profiles of the Monochrome 12 profile, and the compatible scalable format range extensions profiles of the Scalable Monochrome 12 profile
Scalable Monochrome 16	The compatible format range extensions profiles of the Monochrome 16 profile, and the compatible scalable format range extensions profiles of the Scalable Monochrome 16 profile
Scalable Main 4:4:4	Scalable Main, Main, Main Still Picture, the compatible format range extensions profiles of the Main 4:4:4 profile, and the compatible scalable format range extensions profiles of the Scalable Main 4:4:4 profile
Multiview Main	Multiview Main, Main, Main Still Picture
3D Main	3D Main, Multiview Main, Main, Main Still Picture

F.11.3 Derivation of sub-bitstreams subBitstream and baseBitstream

For an output operation point associated with an OLS in a bitstream, let `olsIdx` be the OLS index of the OLS, the sub-bitstream `subBitstream` and base layer sub-bitstream `baseBitstream` are derived as follows:

- The sub-bitstream `subBitstream` is derived as follows:
 - If `OlsIdxToLsIdx[olsIdx]` is less than or equal to `vps_num_layer_sets_minus1`, `subBitstream` is derived by invoking the sub-bitstream extraction process as specified in clause F.10.1 with the following inputs: the bitstream, `tIdTarget` equal to `OpTid` of the output operation point, and `layerIdListTarget` containing the `nuh_layer_id` value `layerId` of the layer and all the reference layers of the layer.
 - Otherwise, `subBitstream` is derived by invoking the sub-bitstream extraction process as specified in clause F.10.3 with `tIdTarget` equal to `OpTid` of the output operation point and with `layerIdListTarget` containing the `nuh_layer_id` value of the layer and all the reference layers of the layer.
- When `vps_base_layer_internal_flag` is equal to 1, the base layer sub-bitstream `baseBitstream` is derived as follows:
 - If VCL NAL units with `nuh_layer_id` equal to 0 are included in `subBitstream`, `baseBitstream` is derived by invoking the sub-bitstream extraction process as specified in clause F.10.1 with the `subBitstream`, `tIdTarget` equal to `OpTid` of the output operation point, and `layerIdListTarget` containing only one `nuh_layer_id` value that is equal to 0 as inputs.
 - Otherwise, `baseBitstream` is derived by invoking the independent non-base layer rewriting process as specified in clause F.10.2 with `subBitstream`, `tIdTarget` equal to `OpTid` of the output operation point, and `layerIdListTarget` containing only the smallest `nuh_layer_id` value of the VCL NAL units of `subBitstream` as inputs.

F.12 Byte stream format

The specifications in Annex B apply.

F.13 Hypothetical reference decoder

F.13.1 General

Three sets of bitstream conformance tests are needed for checking the conformance of a bitstream, which is referred to as the entire bitstream, denoted as `entireBitstream`. The first set of bitstream conformance tests are for testing the conformance of the entire bitstream and its temporal subsets, regardless of whether there is a layer set specified by the active VPS that contains all the `nuh_layer_id` values of VCL NAL units present in the entire bitstream. The second set of bitstream

conformance tests are for testing the conformance of the layer sets specified by the active VPS and their temporal subsets. For all these tests, only the base layer pictures (i.e., pictures with `nuh_layer_id` equal to 0) are decoded and other pictures are ignored by the decoder when the decoding process is invoked.

The first and second sets of bitstream conformance tests are specified in clause C.1. Clause F.13 and its subclauses specify the HRD operations for the third sets of bitstream conformance tests.

The third set of bitstream conformance tests are for testing the conformance of the OLSs specified by the VPS extension part of the active VPS and their temporal subsets. For each test in the third set of bitstream conformance tests, the following ordered steps apply in the order listed, followed by the processes described after these steps in this clause:

1. An output operation point under test, denoted as `TargetOp`, is selected by selecting a value for `TargetOlsIdx` identifying a target OLS and selecting a target highest TemporalId value `HighestTid`. The value of `TargetOlsIdx` shall be in the range of 0 to `NumOutputLayerSets - 1`, inclusive, and the value of `HighestTid` shall be in the range of 0 to `MaxSubLayersInLayerSetMinus1[OlsIdxToLsIdx[TargetOlsIdx]]`, inclusive. Additionally, the values of `TargetOlsIdx` and `HighestTid` are constrained as specified in the next step.

The variables `TargetDecLayerSetIdx`, `TargetOptLayerIdList`, and `TargetDecLayerIdList` are then derived as specified in clause F.8.1.2. The output operation point under test has `OptLayerIdList` equal to `TargetOptLayerIdList`, `OpLayerIdList` equal to `TargetDecLayerIdList` and `OpTid` equal to `HighestTid`.

2. A bitstream to be decoded, `BitstreamToDecode`, is specified as follows:
 - If `TargetDecLayerSetIdx` is less than or equal to `vps_num_layer_sets_minus1` and `vps_base_layer_internal_flag` is equal to 1, the sub-bitstream extraction process as specified in clause 10 is applied with the bitstream `entireBitstream`, `HighestTid` and `TargetDecLayerIdList` as inputs, and the output is assigned to `BitstreamToDecode`.
 - Otherwise, if `TargetDecLayerSetIdx` is less than or equal to `vps_num_layer_sets_minus1` and `vps_base_layer_internal_flag` is equal to 0, the sub-bitstream extraction process as specified in clause F.10.1 is applied with the bitstream `entireBitstream`, `HighestTid` and `TargetDecLayerIdList` as inputs, and the output is assigned to `BitstreamToDecode`.
 - Otherwise, if `TargetDecLayerSetIdx` is greater than `vps_num_layer_sets_minus1`, `NumLayersInIdList[TargetDecLayerSetIdx]` is equal to 1 and the profile of the layer in `TargetOlsIdx` is one of those specified in Annex A, the independent non-base layer rewriting process of clause F.10.2 is applied with the bitstream `entireBitstream`, `HighestTid` and `TargetDecLayerIdList[0]` as inputs, and the output is assigned to `BitstreamToDecode`.
 - Otherwise, the sub-bitstream extraction process as specified in clause F.10.3 is applied with the bitstream `entireBitstream`, `HighestTid` and `TargetDecLayerIdList` as inputs, and the output is assigned to `BitstreamToDecode`.

The values of `TargetOlsIdx` and `HighestTid` are additionally constrained as follows:

- When `vps_base_layer_available_flag` is equal to 0, `OlsIdxToLsIdx[TargetOlsIdx]` shall be in the range of `FirstAddLayerSetIdx` to `LastAddLayerSetIdx`, inclusive.
 - When `vps_base_layer_internal_flag` is equal to 0, `TargetOlsIdx` shall be greater than 0.
 - The value of `TargetOlsIdx` shall be such that there is at least one VCL NAL unit in `BitstreamToDecode` with `nuh_layer_id` equal to `LayerSetLayerIdList[OlsIdxToLsIdx[TargetOlsIdx]][i]` for each value of `i` in the range of 0 to `NumLayersInIdList[OlsIdxToLsIdx[TargetOlsIdx]] - 1`, inclusive.
 - The value of `HighestTid` shall be such that there is at least one VCL NAL unit with TemporalId equal to `HighestTid` in `BitstreamToDecode`.
3. A partitioning scheme is selected from the list of partitioning schemes signalled in the active VPS for the selected OLS. The selected partitioning scheme is denoted as `TargetPartitioningScheme` with partitioning scheme index `TargetPsIdx`.
 4. The subsequent steps apply to each bitstream partition, referred to as the bitstream partition under test `TargetBitstreamPartition`, of the selected partitioning scheme of the target OLS. If there is only one bitstream partition for `TargetPartitioningScheme`, the `TargetBitstreamPartition` is identical to `BitstreamToDecode`. Otherwise, each bitstream partition is derived with the demultiplexing process for deriving a bitstream partition in clause F.13.6, with `BitstreamToDecode`, the list of layers in `TargetBitstreamPartition` and the number of layers in `TargetBitstreamPartition` as inputs.
 5. The applicable `hrd_parameters()` syntax structures and the `sub_layer_hrd_parameters()` syntax structures are selected as follows:
 - A `SchedSelCombIdx` is selected for `BitstreamToDecode` and used for each `TargetBitstreamPartition`. The selected `SchedSelCombIdx` shall be in the range of 0 to `num_bsp_schedules_minus1[TargetOlsIdx][TargetPsIdx][HighestTid]`, inclusive.

- The applicable `hrd_parameters()` syntax structure is the `bsp_hrd_idx[TargetOlsIdx][TargetPsIdx][HighestTid][SchedSelCombIdx][j]`-th `hrd_parameters()` syntax structure in the active VPS, where `j` is the bitstream partition index of `TargetBitstreamPartition`.

Within the selected `hrd_parameters()` syntax structures, if `BitstreamToDecode` is a Type I bitstream, the `sub_layer_hrd_parameters(HighestTid)` syntax structures that immediately follow the condition "`if(vcl_hrd_parameters_present_flag)`" are selected and the variable `NalHrdModeFlag` is set equal to 0; otherwise (`BitstreamToDecode` is a Type II bitstream), the `sub_layer_hrd_parameters(HighestTid)` syntax structures that immediately follow either the condition "`if(vcl_hrd_parameters_present_flag)`" (in this case the variable `NalHrdModeFlag` is set equal to 0) or the condition "`if(nal_hrd_parameters_present_flag)`" (in this case the variable `NalHrdModeFlag` is set equal to 1) are selected. When `BitstreamToDecode` is a Type II bitstream and `NalHrdModeFlag` is equal to 0, all non-VCL NAL units except filler data NAL units, and all leading `zero_8bits`, `zero_byte`, `start_code_prefix_one_3bytes`, and `trailing_zero_8bits` syntax elements that form a byte stream from the NAL unit stream (as specified in Annex B), when present, are discarded from `TargetBitstreamPartition`, and the remaining sub-bitstream is assigned to `TargetBitstreamPartition`.

6. An access unit associated with a buffering period SEI message (present in a bitstream partition nesting SEI message in `BitstreamToDecode` or available through external means not specified in this Specification) applicable to `TargetOp`, `TargetPartitioningScheme` and `TargetBitstreamPartition` is selected as the HRD initialization point and referred to as access unit 0.

The variable `MultiLayerCpbOperationFlag` is derived as follows:

- If `BitstreamToDecode` contains only the base layer (i.e., `TargetOlsIdx` is equal to 0), `MultiLayerCpbOperationFlag` is set equal to 0.
 - Otherwise, `MultiLayerCpbOperationFlag` is set equal to 1.
7. If `sub_pic_hrd_params_present_flag` in the selected `hrd_parameters()` syntax structure is equal to 1, the CPB is scheduled to operate either at the partition unit level (in which case the variable `SubPicHrdFlag` is set equal to 0) or at the sub-partition level (in which case the variable `SubPicHrdFlag` is set equal to 1). Otherwise, `SubPicHrdFlag` is set equal to 0 and the CPB is scheduled to operate at the partition unit level.
 8. For each access unit in `TargetBitstreamPartition` starting from access unit 0, the buffering period SEI message (present in a bitstream partition nesting SEI message in `BitstreamToDecode` or available through external means not specified in this Specification) that is associated with the access unit and applies to `TargetOp`, `TargetPartitioningScheme` and `TargetBitstreamPartition` is selected, the picture timing SEI message (present in a bitstream partition nesting SEI message in `BitstreamToDecode` or available through external means not specified in this Specification) that is associated with the access unit and applies to `TargetOp`, `TargetPartitioningScheme` and `TargetBitstreamPartition` is selected, and when `SubPicHrdFlag` is equal to 1 and `sub_pic_cpb_params_in_pic_timing_sei_flag` is equal to 0, the decoding unit information SEI messages (present in bitstream partition nesting SEI messages in `BitstreamToDecode` or available through external means not specified in this Specification) that are associated with decoding units in the access unit and apply to `TargetOp`, `TargetPartitioningScheme`, and `TargetBitstreamPartition` are selected.
 9. A value of `SchedSelIdx` for `TargetBitstreamPartition` is set equal to `bsp_sched_idx[TargetOlsIdx][TargetPsIdx][HighestTid][SchedSelCombIdx][j]`, where `j` is the index of the bitstream partition index of `TargetBitstreamPartition`.
 10. The variable `initialAltParamSelectionFlag` is derived as follows:
 - If all of the following conditions are true, `initialAltParamSelectionFlag` is set equal to 1:
 - The coded picture with `nuh_layer_id` equal to `SmallestLayerId` in access unit 0 has `nal_unit_type` equal to `CRA_NUT` or `BLA_W_LP`.
 - `MultiLayerCpbOperationFlag` is equal to 0.
 - `irap_cpb_params_present_flag` in the selected buffering period SEI message is equal to 1.
 - Otherwise, if all of the following conditions are true, `initialAltParamSelectionFlag` is set equal to 1:
 - The coded picture with `nuh_layer_id` equal to `SmallestLayerId` in access unit 0 is an IRAP picture.
 - `MultiLayerCpbOperationFlag` is equal to 1.
 - `irap_cpb_params_present_flag` in the selected buffering period SEI message is equal to 1.
 - Otherwise, `initialAltParamSelectionFlag` is set equal to 0.
 11. When `initialAltParamSelectionFlag` is equal to 1, the following applies:
 - The variable `skippedPictureList` is set to consist of the CL-RAS pictures and the RASL pictures associated with the IRAP pictures with `nuh_layer_id` equal to `nuhLayerId` for which `LayerInitializedFlag[nuhLayerId]` is equal to 0 at the start of decoding the IRAP picture and for which `nuhLayerId` is among `TargetDecLayerIdList`.

- Either of the following applies for selection of the initial CPB removal delay and delay offset:
 - If `NalHrdModeFlag` is equal to 1, the default initial CPB removal delay and delay offset represented by `nal_initial_cpb_removal_delay[SchedSelIdx]` and `nal_initial_cpb_removal_offset[SchedSelIdx]`, respectively, in the selected buffering period SEI message are selected. Otherwise, the default initial CPB removal delay and delay offset represented by `vcl_initial_cpb_removal_delay[SchedSelIdx]` and `vcl_initial_cpb_removal_offset[SchedSelIdx]`, respectively, in the selected buffering period SEI message are selected. The variable `DefaultInitCpbParamsFlag` is set equal to 1.
 - If `NalHrdModeFlag` is equal to 0, the alternative initial CPB removal delay and delay offset represented by `nal_initial_alt_cpb_removal_delay[SchedSelIdx]` and `nal_initial_alt_cpb_removal_offset[SchedSelIdx]`, respectively, in the selected buffering period SEI message are selected. Otherwise, the alternative initial CPB removal delay and delay offset represented by `vcl_initial_alt_cpb_removal_delay[SchedSelIdx]` and `vcl_initial_alt_cpb_removal_offset[SchedSelIdx]`, respectively, in the selected buffering period SEI message are selected. The variable `DefaultInitCpbParamsFlag` is set equal to 0 and all the pictures in `skippedPictureList` are discarded from `BitstreamToDecode` and the remaining bitstream is assigned to `BitstreamToDecode`.

Each conformance test consists of a combination of one option in each of the above steps. When there is more than one option for a step, for any particular conformance test only one option is chosen. All possible combinations of all the steps form the entire set of conformance tests.

When `BitstreamToDecode` is a Type II bitstream, the following applies:

- If the `sub_layer_hrd_parameters(HighestTid)` syntax structure that immediately follows the condition "`if(vcl_hrd_parameters_present_flag)`" is selected, the test is conducted at the Type I conformance point and only VCL and filler data NAL units are counted for the input bit rate and CPB storage.
- Otherwise (the `sub_layer_hrd_parameters(HighestTid)` syntax structure that immediately follows the condition "`if(nal_hrd_parameters_present_flag)`" is selected), the test is conducted at the Type II conformance point, and all bytes of the Type II bitstream, which may be a NAL unit stream or a byte stream, are counted for the input bit rate and CPB storage.

NOTE 1 – NAL HRD parameters established by a value of `SchedSelIdx` for the Type II conformance point are sufficient to also establish VCL HRD conformance for the Type I conformance point for the same values of `InitCpbRemovalDelay[SchedSelIdx]`, `BitRate[SchedSelIdx]` and `CpbSize[SchedSelIdx]` for the VBR case (`cbr_flag[SchedSelIdx]` equal to 0). This is because the data flow into the Type I conformance point is a subset of the data flow into the Type II conformance point and because, for the VBR case, the CPB is allowed to become empty and stay empty until the time a next picture is scheduled to begin to arrive. For example, when decoding a CVS conforming to one or more of the profiles specified in Annex G using the decoding process specified in clauses G.2 through G.10, when NAL HRD parameters are provided for the Type II conformance point that not only fall within the bounds set for NAL HRD parameters for profile conformance in item f) of clause G.11.2.2 but also fall within the bounds set for VCL HRD parameters for profile conformance in item e) of clause G.11.2.2, conformance of the VCL HRD for the Type I conformance point is also assured to fall within the bounds of item e) of clause G.11.2.2.

All VPSs, SPSs and PPSs referred to in the VCL NAL units, and the corresponding buffering period, picture timing and decoding unit information SEI messages shall be conveyed to the HRD, in a timely manner, either in the bitstream (by non-VCL NAL units), or by other means not specified in this Specification.

In Annexes C, D and E, the specification for "presence" of non-VCL NAL units that contain VPSs, SPSs, PPSs, buffering period SEI messages, picture timing SEI messages or decoding unit information SEI messages is also satisfied when those NAL units (or just some of them) are conveyed to decoders (or to the HRD) by other means not specified in this Specification. For the purpose of counting bits, only the appropriate bits that are actually present in the bitstream are counted.

NOTE 2 – As an example, synchronization of such a non-VCL NAL unit, conveyed by means other than presence in the bitstream, with the NAL units that are present in the bitstream, can be achieved by indicating two points in the bitstream, between which the non-VCL NAL unit would have been present in the bitstream, had the encoder decided to convey it in the bitstream.

When the content of such a non-VCL NAL unit is conveyed for the application by some means other than presence within the bitstream, the representation of the content of the non-VCL NAL unit is not required to use the same syntax as specified in this Specification.

NOTE 3 – When HRD information is contained within the bitstream, it is possible to verify the conformance of a bitstream to the requirements of this clause based solely on information contained in the bitstream. When the HRD information is not present in the bitstream, as is the case for all "stand-alone" Type I bitstreams, conformance can only be verified when the HRD data are supplied by some other means not specified in this Specification.

For the bitstream-partition-specific CPB operation as specified in this annex, the HRD contains a bitstream demultiplexer (optionally present), two or more bitstream partition buffers (BPB), two or more instantaneous decoding processes, a decoded picture buffer (DPB) that contains a sub-DPB for each layer, and output cropping as shown in Figure F.1.

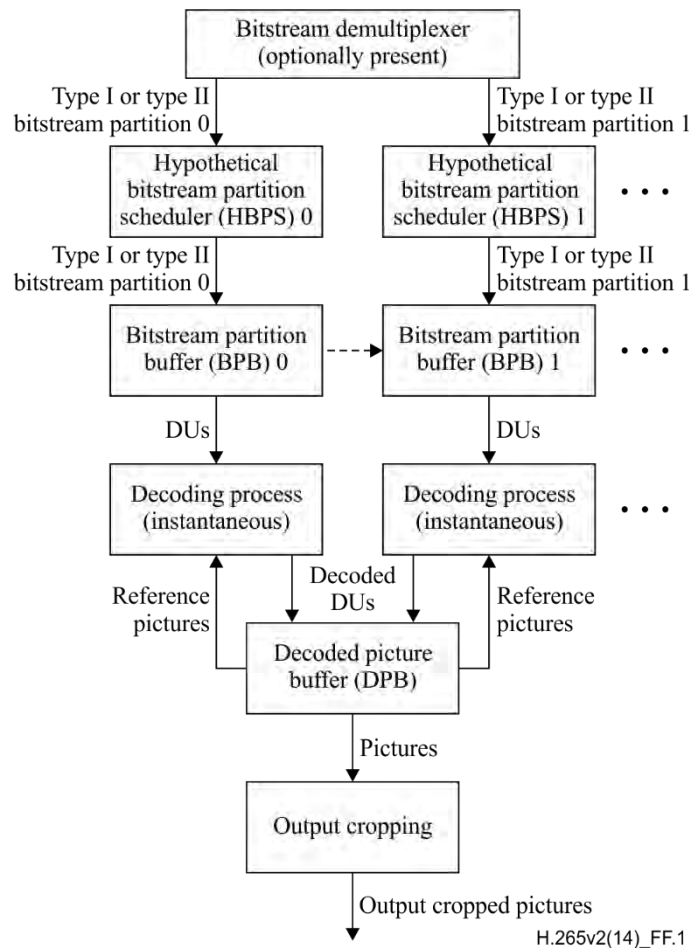


Figure F.1 – Bitstream-partition-specific HRD buffer model

For each bitstream conformance test, the size of BPB (or CPB) $CpbSize[SchedSelIdx]$ is $BpbSize[TargetOlsIdx][TargetPsIdx][HighestTid][SchedSelCombIdx][j]$, and the $BitRate[SchedSelIdx]$ is $BpBitRate[TargetOlsIdx][TargetPsIdx][HighestTid][SchedSelCombIdx][j]$, where j is the bitstream partition index of $TargetBitstreamPartition$, and $SchedSelIdx$ and the HRD parameters are specified above in this clause. The sub-DPB size of the sub-DPB for a layer with nuh_layer_id equal to $currLayerId$ is $max_vps_dec_pic_buffering_minus1[TargetOlsIdx][layerIdx][HighestTid] + 1$, where $layerIdx$ is equal to the value such that $LayerSetLayerIdList[TargetDecLayerSetIdx][layerIdx]$ is equal to $currLayerId$.

If $SubPicHrdFlag$ is equal to 0, the HRD operates at partition unit level and each decoding unit is a partition unit. Otherwise the HRD operates at sub-partition level and each decoding unit is a subset of a partition unit.

NOTE 4 – If the HRD operates at partition unit level, each time when some bits are removed from the CPB, a decoding unit that is an entire partition unit is removed from the CPB. Otherwise (the HRD operates at sub-partition level), each time when some bits are removed from the CPB, a decoding unit that is a subset of a partition unit is removed from the CPB. Regardless of whether the HRD operates at partition unit level or sub-partition level, each time when some picture is output from the DPB, an entire decoded picture is output from the DPB, though the picture output time is derived based on the differently derived CPB removal times and the differently signalled DPB output delays.

The following is specified for expressing the constraints in this annex:

- Each access unit is referred to as access unit n , where the number n identifies the particular access unit. Access unit 0 is selected per step 6 above. The value of n is incremented by 1 for each subsequent access unit in decoding order.
- Each decoding unit is referred to as decoding unit m , where the number m identifies the particular decoding unit. The first decoding unit in decoding order in access unit 0 is referred to as decoding unit 0. The value of m is incremented by 1 for each subsequent decoding unit in decoding order.

NOTE 5 – The numbering of decoding units is relative to the first decoding unit in access unit 0.

- Picture n refers to a particular coded or decoded picture of access unit n .

The HRD operates as follows:

- The HRD is initialized at decoding unit 0, with the CPB, each sub-DPB of the DPB and each BPB being set to be empty (the sub-DPB fullness for each sub-DPB is set equal to 0).
NOTE 6 – After initialization, the HRD is not initialized again by subsequent buffering period SEI messages.
- Data associated with decoding units that flow into the BPB according to a specified arrival schedule are delivered by an HBPS.
- Each bitstream partition with index j is processed as specified in clause F.13.2 with the HSS replaced by the HBPS and with SchedSelIdx equal to $\text{bsp_sched_idx}[\text{TargetOlsIdx}][\text{TargetPsIdx}][\text{HighestTid}][\text{SchedSelCombIdx}][j]$.
- The data associated with each decoding unit are removed and decoded instantaneously by the instantaneous decoding process at the CPB removal time of the decoding unit.
- Each decoded picture is placed in the DPB.
- A decoded picture is removed from the DPB when it becomes no longer needed for inter prediction reference and no longer needed for output.

For each bitstream conformance test, the operation of the CPB and the BPB is specified in clause F.13.2, the instantaneous decoder operation is specified in clauses 2 through 10, clauses F.2 through F.10, and either clauses G.2 through G.10 or clauses H.2 through H.10, the operation of the DPB is specified in clause F.13.3, and the output cropping is specified in clauses F.13.3.3 and F.13.5.2.2.

HSS, HBPS and HRD information concerning the number of enumerated delivery schedules and their associated bit rates and buffer sizes is specified in clauses E.2.2, E.2.3, F.7.3.2.1.6 and F.7.4.3.1.6. The HRD is initialized as specified by the buffering period SEI message specified in clauses D.2.2 and D.3.2. The removal timing of decoding units from the CPB and output timing of decoded pictures from the DPB is specified using information in picture timing SEI messages (specified in clauses D.2.3 and D.3.3) or in decoding unit information SEI messages (specified in clauses D.2.22 and D.3.22). All timing information relating to a specific decoding unit shall arrive prior to the CPB removal time of the decoding unit.

The requirements for bitstream conformance are specified in clause F.13.4 and the HRD is used to check conformance of bitstreams as specified above in this clause and to check conformance of decoders as specified in clause F.13.5.

F.13.2 Operation of bitstream partition buffer

F.13.2.1 General

The specifications in this clause apply independently to each set of bitstream partition buffer (BPB) parameters that is present and to both the Type I and Type II conformance points and the set of BPB parameters is selected as specified in clause F.13.1. In clause F.13.2 and its subclauses, CPB is understood to be BPB and access unit is understood to be partition unit.

F.13.2.2 Timing of decoding unit arrival

The variable altParamSelectionFlag is derived as follows:

- If all of the following conditions are true, altParamSelectionFlag is set equal to 1:
 - The current picture is a BLA picture that has nal_unit_type equal to BLA_W_LP and nuh_layer_id equal to SmallestLayerId or is a CRA picture that has nuh_layer_id equal to SmallestLayerId.
 - MultiLayerCpbOperationFlag is equal to 0.
- Otherwise, if all of the following conditions are true, altParamSelectionFlag is set equal to 1:
 - The current picture is an IRAP picture with nuh_layer_id equal to SmallestLayerId and with NoClrasOutputFlag equal to 1.
 - MultiLayerCpbOperationFlag is equal to 1.
- Otherwise, altParamSelectionFlag is set equal to 0.

When altParamSelectionFlag is equal to 1, the following applies:

- If some external means not specified in this Specification is available to set the variable UseAltCpbParamsFlag to a value, UseAltCpbParamsFlag is set equal to the value provided by the external means.
- Otherwise, UseAltCpbParamsFlag is set equal to the value of use_alt_cpb_params_flag of the buffering period SEI message selected as specified in clause F.13.1.

If SubPicHrdFlag is equal to 0, the variable subPicParamsFlag is set equal to 0 and the process specified in the remainder of this clause is invoked with a decoding unit being considered as an access unit, for derivation of the initial and final CPB arrival times for access unit n.

Otherwise (SubPicHrdFlag is equal to 1), the process specified in the remainder of this clause is first invoked with the variable subPicParamsFlag set equal to 0 and a decoding unit being considered as an access unit, for derivation of the initial and final CPB arrival times for access unit n, and then invoked with subPicParamsFlag set equal to 1 and a decoding unit being considered as a subset of an access unit, for derivation of the initial and final CPB arrival times for the decoding units in access unit n.

The variables InitCpbRemovalDelay[SchedSelIdx] and InitCpbRemovalDelayOffset[SchedSelIdx] are derived as follows:

- If one or more of the following conditions are true, InitCpbRemovalDelay[SchedSelIdx] and InitCpbRemovalDelayOffset[SchedSelIdx] are set equal to the values of the buffering period SEI message syntax elements nal_initial_alt_cpb_removal_delay[SchedSelIdx] and nal_initial_alt_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 1, or vcl_initial_alt_cpb_removal_delay[SchedSelIdx] and vcl_initial_alt_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 0, where the buffering period SEI message syntax elements are selected as specified in clause F.13.1:
 - Access unit 0 includes a BLA picture with nuh_layer_id equal to SmallestLayerId and nal_unit_type equal to BLA_W_RADL or BLA_N_LP, MultiLayerCpbOperationFlag is equal to 0 and the value of irap_cpb_params_present_flag of the buffering period SEI message is equal to 1.
 - Access unit 0 includes a BLA picture with nuh_layer_id equal to SmallestLayerId and nal_unit_type equal to BLA_W_LP or includes a CRA picture with nuh_layer_id equal to SmallestLayerId, MultiLayerCpbOperationFlag is equal to 0, and the value of irap_cpb_params_present_flag of the buffering period SEI message is equal to 1 and one or more of the following conditions are true:
 - UseAltCpbParamsFlag for access unit 0 is equal to 1.
 - DefaultInitCpbParamsFlag is equal to 0.
 - Access unit 0 includes an IRAP picture with nuh_layer_id equal to SmallestLayerId, MultiLayerCpbOperationFlag is equal to 1 and the value of irap_cpb_params_present_flag of the buffering period SEI message is equal to 1 and one or more of the following conditions are true:
 - UseAltCpbParamsFlag for access unit 0 is equal to 1.
 - DefaultInitCpbParamsFlag is equal to 0.
 - The value of subPicParamsFlag is equal to 1.
- Otherwise, InitCpbRemovalDelay[SchedSelIdx] and InitCpbRemovalDelayOffset[SchedSelIdx] are set equal to the values of the buffering period SEI message syntax elements nal_initial_cpb_removal_delay[SchedSelIdx] and nal_initial_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 1, or vcl_initial_cpb_removal_delay[SchedSelIdx] and vcl_initial_cpb_removal_offset[SchedSelIdx], respectively, when NalHrdModeFlag is equal to 0, where the buffering period SEI message syntax elements are selected as specified in clause F.13.1.

The time at which the first bit of decoding unit m begins to enter the CPB is referred to as the initial arrival time initArrivalTime[m].

Decoding units are indexed in decoding order within the bitstream.

The initial arrival time of decoding unit m is derived as follows:

- If the decoding unit is decoding unit 0 (i.e., when m is equal to 0) and the decoding unit belongs to the base bitstream partition, initArrivalTime[0] is set equal to 0.
- Otherwise, if the decoding unit is decoding unit 0 and the decoding unit does not belong to the base bitstream partition, initArrivalTime[0] is obtained from the bitstream partition initial arrival time SEI message that applies to TargetOp, TargetPartitioningScheme, and TargetBitstreamPartition and is associated with the access unit containing decoding unit 0. The applicable bitstream partition initial arrival time SEI message is either included in a bitstream partition nesting SEI message in an SEI NAL unit in the access unit containing decoding unit 0 in BitstreamToDecode or available through external means not specified in this Specification. If NalHrdModeFlag is equal to 0, initArrivalTime[0] is set equal to vcl_initial_arrival_delay[SchedSelIdx] in the applicable bitstream partition initial arrival time SEI message. Otherwise (NalHrdModeFlag is equal to 1), initArrivalTime[0] is set equal to nal_initial_arrival_delay[SchedSelIdx] in the applicable bitstream partition initial arrival time SEI message.
- Otherwise, the following applies:

- If $cbr_flag[SchedSelIdx]$ is equal to 1, the initial arrival time for decoding unit m is equal to the final arrival time (which is derived below) of decoding unit $m - 1$, i.e.,

$$\begin{aligned} & \text{if(!subPicParamsFlag)} \\ & \quad \text{initArrivalTime}[m] = \text{AuFinalArrivalTime}[m - 1] \\ & \text{else} \\ & \quad \text{initArrivalTime}[m] = \text{DuFinalArrivalTime}[m - 1] \end{aligned} \quad (\text{F-69})$$

- Otherwise ($cbr_flag[SchedSelIdx]$ is equal to 0), the initial arrival time for decoding unit m is derived as follows:

$$\begin{aligned} & \text{if(!subPicParamsFlag)} \\ & \quad \text{initArrivalTime}[m] = \text{Max}(\text{AuFinalArrivalTime}[m - 1], \text{initArrivalEarliestTime}[m]) \\ & \quad (\text{F-70}) \\ & \text{else} \\ & \quad \text{initArrivalTime}[m] = \text{Max}(\text{DuFinalArrivalTime}[m - 1], \text{initArrivalEarliestTime}[m]) \end{aligned}$$

where $\text{initArrivalEarliestTime}[m]$ is derived as follows:

- The variable $\text{tmpNominalRemovalTime}$ is derived as follows:

$$\begin{aligned} & \text{if(!subPicParamsFlag)} \\ & \quad \text{tmpNominalRemovalTime} = \text{AuNominalRemovalTime}[m] \\ & \text{else} \\ & \quad \text{tmpNominalRemovalTime} = \text{DuNominalRemovalTime}[m] \end{aligned} \quad (\text{F-71})$$

where $\text{AuNominalRemovalTime}[m]$ and $\text{DuNominalRemovalTime}[m]$ are the nominal CPB removal time of access unit m and decoding unit m , respectively, as specified in clause F.13.2.3.

- If decoding unit m is not the first decoding unit of a subsequent buffering period, $\text{initArrivalEarliestTime}[m]$ is derived as follows:

$$\begin{aligned} \text{initArrivalEarliestTime}[m] = & \text{tmpNominalRemovalTime} - \\ & (\text{InitCpbRemovalDelay}[SchedSelIdx] \\ & + \text{InitCpbRemovalDelayOffset}[SchedSelIdx]) \div 90\,000 \end{aligned} \quad (\text{F-72})$$

- Otherwise (decoding unit m is the first decoding unit of a subsequent buffering period), $\text{initArrivalEarliestTime}[m]$ is derived as follows:

$$\begin{aligned} \text{initArrivalEarliestTime}[m] = & \text{tmpNominalRemovalTime} - \\ & (\text{InitCpbRemovalDelay}[SchedSelIdx] \div 90\,000) \end{aligned} \quad (\text{F-73})$$

The final arrival time for decoding unit m is derived as follows:

$$\begin{aligned} & \text{if(!subPicParamsFlag)} \\ & \quad \text{AuFinalArrivalTime}[m] = \text{initArrivalTime}[m] + \text{sizeInbits}[m] \div \text{BitRate}[SchedSelIdx] \\ & \quad (\text{F-74}) \\ & \text{else} \\ & \quad \text{DuFinalArrivalTime}[m] = \text{initArrivalTime}[m] + \text{sizeInbits}[m] \div \text{BitRate}[SchedSelIdx] \end{aligned}$$

where $\text{sizeInbits}[m]$ is the size in bits of decoding unit m , counting the bits of the VCL NAL units and the filler data NAL units for the Type I conformance point or all bits of the Type II bitstream for the Type II conformance point.

The values of $SchedSelIdx$, $\text{BitRate}[SchedSelIdx]$ and $\text{CpbSize}[SchedSelIdx]$ are constrained as follows:

- If the lists of delivery schedules, including the HRD parameters associated with the individual delivery schedules, for the access unit containing decoding unit m and the previous access unit differ, the HSS selects a value $SchedCombIdx1$ in the range of 0 to $\text{num_bsp_schedules_minus1}[\text{TargetOlsIdx}][\text{TargetPsIdx}][\text{HighestTid}]$, inclusive, and sets $SchedSelIdx1$ for $\text{TargetBitstreamPartition}$ to be equal to $\text{bsp_sched_idx}[\text{TargetOlsIdx}][\text{TargetPsIdx}][\text{HighestTid}][\text{SchedSelCombIdx}][j]$, where j is the index of the bitstream partition index of $\text{TargetBitstreamPartition}$, for the access unit containing decoding unit m that results in a $\text{BitRate}[SchedSelIdx1]$ or $\text{CpbSize}[SchedSelIdx1]$ for the access unit containing decoding unit m . The value of $\text{BitRate}[SchedSelIdx1]$ or $\text{CpbSize}[SchedSelIdx1]$ may differ from the value of $\text{BitRate}[SchedSelIdx0]$ or $\text{CpbSize}[SchedSelIdx0]$ for the value $SchedSelIdx0$ of $SchedSelIdx$ that was in use for the previous access unit.
- Otherwise, the HSS continues to operate with the previous values of $SchedSelIdx$, $\text{BitRate}[SchedSelIdx]$ and $\text{CpbSize}[SchedSelIdx]$.

When the HSS selects values of `BitRate[SchedSelIdx]` or `CpbSize[SchedSelIdx]` that differ from those of the previous access unit, the following applies:

- The variable `BitRate[SchedSelIdx]` comes into effect at the initial CPB arrival time of the current access unit.
- The variable `CpbSize[SchedSelIdx]` comes into effect as follows:
 - If the new value of `CpbSize[SchedSelIdx]` is greater than the old CPB size, it comes into effect at the initial CPB arrival time of the current access unit.
 - Otherwise, the new value of `CpbSize[SchedSelIdx]` comes into effect at the CPB removal time of the current access unit.

F.13.2.3 Timing of decoding unit removal and decoding of decoding unit

The variables `InitCpbRemovalDelay[SchedSelIdx]`, `InitCpbRemovalDelayOffset[SchedSelIdx]`, `CpbDelayOffset` and `DpbDelayOffset` are derived as follows:

- If one or more of the following conditions are true, `CpbDelayOffset` is set equal to the value of the buffering period SEI message syntax element `cpb_delay_offset`, `DpbDelayOffset` is set equal to the value of the buffering period SEI message syntax element `dpb_delay_offset` and `InitCpbRemovalDelay[SchedSelIdx]` and `InitCpbRemovalDelayOffset[SchedSelIdx]` are set equal to the values of the buffering period SEI message syntax elements `nal_initial_alt_cpb_removal_delay[SchedSelIdx]` and `nal_initial_alt_cpb_removal_offset[SchedSelIdx]`, respectively, when `NalHrdModeFlag` is equal to 1, or `vcl_initial_alt_cpb_removal_delay[SchedSelIdx]` and `vcl_initial_alt_cpb_removal_offset[SchedSelIdx]`, respectively, when `NalHrdModeFlag` is equal to 0, where the buffering period SEI message containing the syntax elements is selected as specified in clause F.13.1:
 - Access unit 0 includes a BLA picture with `nuh_layer_id` equal to `SmallestLayerId` and `nal_unit_type` equal to `BLA_W_RADL` or `BLA_N_LP`, `MultiLayerCpbOperationFlag` is equal to 0 and the value of `irap_cpb_params_present_flag` of the buffering period SEI message is equal to 1.
 - Access unit 0 includes a BLA picture with `nuh_layer_id` equal to `SmallestLayerId` and `nal_unit_type` equal to `BLA_W_LP` or includes a CRA picture with `nuh_layer_id` equal to `SmallestLayerId`, `MultiLayerCpbOperationFlag` is equal to 0 and the value of `irap_cpb_params_present_flag` of the buffering period SEI message is equal to 1, and one or more of the following conditions are true:
 - `UseAltCpbParamsFlag` for access unit 0 is equal to 1.
 - `DefaultInitCpbParamsFlag` is equal to 0.
 - Access unit 0 includes an IRAP picture with `nuh_layer_id` equal to `SmallestLayerId`, `MultiLayerCpbOperationFlag` is equal to 1 and the value of `irap_cpb_params_present_flag` of the buffering period SEI message is equal to 1 and one or more of the following conditions are true:
 - `UseAltCpbParamsFlag` for access unit 0 is equal to 1.
 - `DefaultInitCpbParamsFlag` is equal to 0.
- Otherwise, `InitCpbRemovalDelay[SchedSelIdx]` and `InitCpbRemovalDelayOffset[SchedSelIdx]` are set equal to the values of the buffering period SEI message syntax elements `nal_initial_cpb_removal_delay[SchedSelIdx]` and `nal_initial_cpb_removal_offset[SchedSelIdx]`, respectively, when `NalHrdModeFlag` is equal to 1, or `vcl_initial_cpb_removal_delay[SchedSelIdx]` and `vcl_initial_cpb_removal_offset[SchedSelIdx]`, respectively, when `NalHrdModeFlag` is equal to 0, where the buffering period SEI message containing the syntax elements is selected as specified in clause F.13.1, `CpbDelayOffset` and `DpbDelayOffset` are both set equal to 0.

The nominal removal time of the access unit `n` from the CPB is specified as follows:

- If access unit `n` is the access unit with `n` equal to 0 (the access unit that initializes the HRD), the nominal removal time of the access unit from the CPB is specified by:

$$AuNominalRemovalTime[0] = InitCpbRemovalDelay[SchedSelIdx] \div 90\,000 \quad (F-75)$$

- Otherwise, the following applies:
 - When access unit `n` is the first access unit of a buffering period that does not initialize the HRD, the following applies:

The nominal removal time of the access unit `n` from the CPB is specified by:

```

if( !concatenationFlag ) {
    baseTime = AuNominalRemovalTime[ firstPicInPrevBuffPeriod ]
    tmpCpbRemovalDelay = AuCpbRemovalDelayVal

```



```

} else {
    baseTime = AuNominalRemovalTime[ prevNonDiscardablePic ]
    tmpCpbRemovalDelay =
        Max( ( auCpbRemovalDelayDeltaMinus1 + 1 ),
            Ceil( ( InitCpbRemovalDelay[ SchedSelIdx ] ÷ 90 000 +
                AuFinalArrivalTime[ n - 1 ] - AuNominalRemovalTime[ n - 1 ] ) ÷ ClockTick ) )
}
AuNominalRemovalTime[ n ] = baseTime + ClockTick * ( tmpCpbRemovalDelay -
    CpbDelayOffset )

```

(F-76)

where $AuNominalRemovalTime[firstPicInPrevBuffPeriod]$ is the nominal removal time of the first access unit of the previous buffering period, $AuNominalRemovalTime[prevNonDiscardablePic]$ is the nominal removal time of the preceding access unit in decoding order, each picture of which is with TemporalId equal to 0 that is not a RASL, RADL or SLNR picture, $AuCpbRemovalDelayVal$ is the value of $AuCpbRemovalDelayVal$ derived according to $au_cpb_removal_delay_minus1$ in the picture timing SEI message, selected as specified in clause F.13.1, associated with access unit n , and concatenationFlag and $auCpbRemovalDelayDeltaMinus1$ are the values of the syntax elements concatenation_flag and $au_cpb_removal_delay_delta_minus1$, respectively, in the buffering period SEI message, selected as specified in clause F.13.1, associated with access unit n .

After the derivation of the nominal CPB removal time and before the derivation of the DPB output time of access unit n , the values of CpbDelayOffset and DpbDelayOffset are updated as follows:

- If one or more of the following conditions are true, CpbDelayOffset is set equal to the value of the buffering period SEI message syntax element cpb_delay_offset and DpbDelayOffset is set equal to the value of the buffering period SEI message syntax element dpb_delay_offset , where the buffering period SEI message containing the syntax elements is selected as specified in clause F.13.1:
 - Access unit n includes a BLA picture with nuh_layer_id equal to SmallestLayerId and nal_unit_type equal to BLA_W_RADL or BLA_N_LP, MultiLayerCpbOperationFlag is equal to 0 and the value of $irap_cpb_params_present_flag$ of the buffering period SEI message is equal to 1.
 - Access unit n includes a BLA picture with nuh_layer_id equal to SmallestLayerId and nal_unit_type equal to BLA_W_LP or includes a CRA picture with nuh_layer_id equal to SmallestLayerId, MultiLayerCpbOperationFlag is equal to 0, the value of $irap_cpb_params_present_flag$ of the buffering period SEI message is equal to 1 and UseAltCpbParamsFlag for access unit n is equal to 1.
 - Access unit n includes an IRAP picture with nuh_layer_id equal to SmallestLayerId, MultiLayerCpbOperationFlag is equal to 1, the value of $irap_cpb_params_present_flag$ of the buffering period SEI message is equal to 1 and UseAltCpbParamsFlag for access unit n is equal to 1.
- Otherwise, CpbDelayOffset and DpbDelayOffset are both set equal to 0.
- When access unit n is not the first access unit of a buffering period, the nominal removal time of the access unit n from the CPB is specified by:

$$AuNominalRemovalTime[n] = AuNominalRemovalTime[firstPicInCurrBuffPeriod] + ClockTick * (AuCpbRemovalDelayVal - CpbDelayOffset) \quad (F-77)$$

where $AuNominalRemovalTime[firstPicInCurrBuffPeriod]$ is the nominal removal time of the first access unit of the current buffering period, and $AuCpbRemovalDelayVal$ is the value of $AuCpbRemovalDelayVal$ derived according to $au_cpb_removal_delay_minus1$ in the picture timing SEI message, selected as specified in clause F.13.1, associated with access unit n .

When SubPicHrdFlag is equal to 1, the following applies:

- The variable $duCpbRemovalDelayInc$ is derived as follows:
 - If $sub_pic_cpb_params_in_pic_timing_sei_flag$ is equal to 0, $duCpbRemovalDelayInc$ is set equal to the value of $du_spt_cpb_removal_delay_increment$ in the decoding unit information SEI message, selected as specified in clause F.13.1, associated with decoding unit m .
 - Otherwise, if $du_common_cpb_removal_delay_flag$ is equal to 0, $duCpbRemovalDelayInc$ is set equal to the value of $du_cpb_removal_delay_increment_minus1[i] + 1$ for decoding unit m in the picture timing SEI message, selected as specified in clause F.13.1, associated with access unit n , where the value of i is 0 for the first $num_nalus_in_du_minus1[0] + 1$ consecutive NAL units in the access unit that contains decoding unit m , 1 for the subsequent $num_nalus_in_du_minus1[1] + 1$ NAL units in the same access unit, 2 for the subsequent $num_nalus_in_du_minus1[2] + 1$ NAL units in the same access unit, etc.

- Otherwise, `duCpbRemovalDelayInc` is set equal to the value of `du_common_cpb_removal_delay_increment_minus1 + 1` in the picture timing SEI message, selected as specified in clause F.13.1, associated with access unit `n`.
- The nominal removal time of decoding unit `m` from the CPB is specified as follows, where `AuNominalRemovalTime[n]` is the nominal removal time of access unit `n`:
 - If decoding unit `m` is the last decoding unit in access unit `n`, the nominal removal time of decoding unit `m` `DuNominalRemovalTime[m]` is set equal to `AuNominalRemovalTime[n]`.
 - Otherwise (decoding unit `m` is not the last decoding unit in access unit `n`), the nominal removal time of decoding unit `m` `DuNominalRemovalTime[m]` is derived as follows:

$$\begin{aligned}
 &\text{if(sub_pic_cpb_params_in_pic_timing_sei_flag)} \\
 &\quad \text{DuNominalRemovalTime[m]} = \text{DuNominalRemovalTime[m + 1]} - \\
 &\quad \text{ClockSubTick} * \text{duCpbRemovalDelayInc} \\
 &\text{else} \\
 &\quad \text{DuNominalRemovalTime[m]} = \text{AuNominalRemovalTime[n]} - \\
 &\quad \text{ClockSubTick} * \text{duCpbRemovalDelayInc}
 \end{aligned}
 \tag{F-78}$$

If `SubPicHrdFlag` is equal to 0, the removal time of access unit `n` from the CPB is specified as follows, where `AuFinalArrivalTime[n]` and `AuNominalRemovalTime[n]` are the final CPB arrival time and nominal CPB removal time, respectively, of access unit `n`:

$$\begin{aligned}
 &\text{if(!low_delay_hrd_flag[HighestTid] || AuNominalRemovalTime[n] >=} \\
 &\quad \text{AuFinalArrivalTime[n]}) \\
 &\quad \text{AuCpbRemovalTime[n]} = \text{AuNominalRemovalTime[n]} \\
 &\text{else} \\
 &\quad \text{AuCpbRemovalTime[n]} = \text{AuNominalRemovalTime[n]} + \text{ClockTick} * \\
 &\quad \text{Ceil((AuFinalArrivalTime[n] - AuNominalRemovalTime[n]) } \div \text{ClockTick})
 \end{aligned}
 \tag{F-79}$$

NOTE 1 – When `low_delay_hrd_flag[HighestTid]` is equal to 1 and `AuNominalRemovalTime[n]` is less than `AuFinalArrivalTime[n]`, the size of access unit `n` is so large that it prevents removal at the nominal removal time.

Otherwise (`SubPicHrdFlag` is equal to 1), the removal time of decoding unit `m` from the CPB is specified as follows:

$$\begin{aligned}
 &\text{– The following applies:} \\
 &\quad \text{if(!low_delay_hrd_flag[HighestTid] || DuNominalRemovalTime[m] >=} \\
 &\quad \quad \text{DuFinalArrivalTime[m]}) \\
 &\quad \quad \text{DuCpbRemovalTime[m]} = \text{DuNominalRemovalTime[m]} \\
 &\quad \text{else} \\
 &\quad \quad \text{DuCpbRemovalTime[m]} = \text{DuFinalArrivalTime[m]}
 \end{aligned}
 \tag{F-80}$$

NOTE 2 – When `low_delay_hrd_flag[HighestTid]` is equal to 1 and `DuNominalRemovalTime[m]` is less than `DuFinalArrivalTime[m]`, the size of decoding unit `m` is so large that it prevents removal at the nominal removal time.

- When `cbr_flag[SchedSelIdx]` is equal to 0, the following applies:
 - Let `refDuCpbRemovalTime` be equal to the CPB removal time of the previous DU preceding the current DU in decoding order (regardless of the bitstream partitions to which the previous DU and the current DU belong).
 - The variable `DuCpbRemovalTime[m]` is modified as follows:

$$\text{DuCpbRemovalTime[m]} = \text{Max(DuCpbRemovalTime[m], refDuCpbRemovalTime)}
 \tag{F-81}$$

If `SubPicHrdFlag` is equal to 0, at the CPB removal time of access unit `n`, the access unit is instantaneously decoded.

Otherwise (`SubPicHrdFlag` is equal to 1), at the CPB removal time of decoding unit `m`, the decoding unit is instantaneously decoded, and when decoding unit `m` is the last decoding unit of access unit `n`, the following applies:

- Access unit `n` is considered as decoded.
- The final CPB arrival time of access unit `n`, i.e., `AuFinalArrivalTime[n]`, is set equal to the final CPB arrival time of the last decoding unit in access unit `n`, i.e., `DuFinalArrivalTime[m]`.
- The nominal CPB removal time of access unit `n`, i.e., `AuNominalRemovalTime[n]`, is set equal to the nominal CPB removal time of the last decoding unit in access unit `n`, i.e., `DuNominalRemovalTime[m]`.

- The CPB removal time of access unit n , i.e., $AuCpbRemovalTime[m]$, is set equal to the CPB removal time of the last decoding unit in access unit n , i.e., $DuCpbRemovalTime[m]$.

F.13.3 Operation of decoded picture buffer

F.13.3.1 General

The specifications in this clause apply independently to each set of decoded picture buffer (DPB) parameters selected as specified in clause F.13.1.

The decoded picture buffer consists of sub-DPBs and each sub-DPB contains picture storage buffers for storage of decoded pictures of one layer. Each of the picture storage buffers of a sub-DPB may contain a decoded picture that is marked as "used for reference" or is held for future output.

The processes specified in clauses F.13.3.2, F.13.3.3, F.13.3.4 and F.13.3.5 are sequentially applied as specified below, and are applied independently for each layer, starting from the base layer, in increasing order of nuh_layer_id values of the layers in the bitstream. When these processes are applied for a particular layer, only the sub-DPB for the particular layer is affected. In the descriptions of these processes, the DPB refers to the sub-DPB for the particular layer, and the particular layer is referred to as the current layer.

NOTE – In the operation of output timing DPB, decoded pictures with $PicOutputFlag$ equal to 1 in the same access unit are output consecutively in ascending order of the nuh_layer_id values of the decoded pictures.

Let picture n and the current picture be the coded picture or decoded picture of the access unit n for a particular value of nuh_layer_id , wherein n is a non-negative integer number.

F.13.3.2 Removal of pictures from the DPB before decoding of the current picture

When the current picture is not picture 0 in the current layer, the removal of pictures in the current layer, with nuh_layer_id equal to $currLayerId$, from the DPB before decoding of the current picture, i.e., picture n , but after parsing the slice header of the first slice of the current picture, happens instantaneously at the CPB removal time of the first decoding unit of the current picture and proceeds as follows:

- The decoding process for RPS as specified in clause F.8.3.2 is invoked.
- The variable $listOfSubDpbsToEmpty$ is derived as follows:
 - If a new VPS is activated by the current access unit or the current picture is an IRAP picture with nuh_layer_id equal to $SmallestLayerId$, $NoRaslOutputFlag$ equal to 1 and $NoCllasOutputFlag$ equal to 1, $listOfSubDpbsToEmpty$ is set to include all the sub-DPBs.
 - Otherwise, if the current picture is an IRAP picture with any nuh_layer_id value $indepLayerId$ such that $NumDirectRefLayers[indepLayerId]$ is equal to 0 and $indepLayerId$ is greater than $SmallestLayerId$, and with $NoRaslOutputFlag$ equal to 1, and $LayerResetFlag$ is equal to 1, $listOfSubDpbsToEmpty$ is set equal to the sub-DPBs containing the current layer and the sub-DPBs containing the predicted layers of the current layer.
 - Otherwise, $listOfSubDpbsToEmpty$ is set equal to the sub-DPB containing the current layer.
- When the current picture is an IRAP picture with $NoRaslOutputFlag$ equal to 1 and any of the following conditions is true:
 - nuh_layer_id equal to $SmallestLayerId$,
 - nuh_layer_id of the current layer is greater than $SmallestLayerId$ and $NumDirectRefLayers[nuh_layer_id]$ is equal to 0,

the following ordered steps are applied:

1. The variable $NoOutputOfPriorPicsFlag$ is derived for the decoder under test as follows:
 - If the current picture is a CRA picture, $NoOutputOfPriorPicsFlag$ is set equal to 1 (regardless of the value of $no_output_of_prior_pics_flag$).
 - Otherwise, if the value of $pic_width_in_luma_samples$, $pic_height_in_luma_samples$, $chroma_format_idc$, $bit_depth_luma_minus8$, $bit_depth_chroma_minus8$, $separate_colour_plane_flag$ or $sps_max_dec_pic_buffering_minus1[HighestTid]$ derived from the active SPS for the current layer is different from the value of $pic_width_in_luma_samples$, $pic_height_in_luma_samples$, $chroma_format_idc$, $bit_depth_luma_minus8$, $bit_depth_chroma_minus8$, $separate_colour_plane_flag$ or $sps_max_dec_pic_buffering_minus1[HighestTid]$, respectively, derived from the SPS that was active for the current layer when decoding the preceding picture in the current layer, $NoOutputOfPriorPicsFlag$ may (but should not) be set equal to 1 by the decoder under test, regardless of the value of $no_output_of_prior_pics_flag$.

NOTE – Although setting NoOutputOfPriorPicsFlag equal to no_output_of_prior_pics_flag is preferred under these conditions, the decoder under test is allowed to set NoOutputOfPriorPicsFlag to 1 in this case.

- Otherwise, NoOutputOfPriorPicsFlag is set equal to no_output_of_prior_pics_flag.
- 2. When the value of NoOutputOfPriorPicsFlag derived for the decoder under test is equal to 1, all non-empty picture storage buffers in all the sub-DPBs contained in listOfSubDpbsToEmpty are emptied without output of the pictures they contain, and the sub-DPB fullness of each sub-DPB in listOfSubDpbsToEmpty is set equal to 0.
- When both of the following conditions are true for any pictures k in the DPB, all such pictures k in the DPB are removed from the DPB:
 - picture k is marked as "unused for reference".
 - picture k has PicOutputFlag equal to 0 or its DPB output time is less than or equal to the CPB removal time of the first decoding unit (denoted as decoding unit m) of the current picture n; i.e., DpbOutputTime[k] is less than or equal to DuCpbRemovalTime[m].
- For each picture that is removed from the DPB, the DPB fullness is decremented by one.

F.13.3.3 Picture output

The processes specified in this clause happen instantaneously at the CPB removal time of access unit n, AuCpbRemovalTime[n].

When access unit n has AuOutputFlag equal to 1, its DPB output time DpbOutputTime[n] is derived as follows, where the variable firstPicInBufferingPeriodFlag is equal to 1 if access unit n is the first access unit of a buffering period and 0 otherwise:

```

if( !SubPicHrdFlag ) {
    DpbOutputTime[ n ] = AuCpbRemovalTime[ n ] + ClockTick * picDpbOutputDelay      (F-82)
    if( firstPicInBufferingPeriodFlag )
        DpbOutputTime[ n ] -= ClockTick * DpbDelayOffset
} else
    DpbOutputTime[ n ] = AuCpbRemovalTime[ n ] + ClockSubTick * picSptDpbOutputDuDelay

```

where picDpbOutputDelay is the value of pic_dpb_output_delay in the picture timing SEI message associated with access unit n, and picSptDpbOutputDuDelay is the value of pic_spt_dpb_output_du_delay, when present, in the decoding unit information SEI messages associated with access unit n, or the value of pic_dpb_output_du_delay in the picture timing SEI message associated with access unit n when there is no decoding unit information SEI message associated with access unit n or no decoding unit information SEI message associated with access unit n has pic_spt_dpb_output_du_delay present.

NOTE – When the syntax element pic_spt_dpb_output_du_delay is not present in any decoding unit information SEI message associated with access unit n, the value is inferred to be equal to pic_dpb_output_du_delay in the picture timing SEI message associated with access unit n.

The output of the current picture contained in access unit n is specified as follows:

- If PicOutputFlag is equal to 1 and DpbOutputTime[n] is equal to AuCpbRemovalTime[n], the current picture is output.
- Otherwise, if PicOutputFlag is equal to 0, the current picture is not output, but will be stored in the DPB as specified in clause F.13.3.4.
- Otherwise (PicOutputFlag is equal to 1 and DpbOutputTime[n] is greater than AuCpbRemovalTime[n]), the current picture is output later and will be stored in the DPB (as specified in clause F.13.3.4) and is output at time DpbOutputTime[n] unless indicated not to be output by NoOutputOfPriorPicsFlag equal to 1.

When output, a picture is cropped, using the conformance cropping window specified in the active SPS for the layer containing the picture.

When access unit n is an access unit that has AuOutputFlag equal to 1 and is not the last access unit of the bitstream that has AuOutputFlag equal to 1, the value of the variable DpbOutputInterval[n] is derived as follows:

$$\text{DpbOutputInterval}[n] = \text{DpbOutputTime}[\text{nextAuInOutputOrder}] - \text{DpbOutputTime}[n] \quad (\text{F-83})$$

where nextAuInOutputOrder is the access unit that follows access unit n in output order and has AuOutputFlag equal to 1.

F.13.3.4 Current decoded picture marking and storage

The current decoded picture after the invocation of the in-loop filter process as specified in clause F.8.7 is stored in the DPB in an empty picture storage buffer, the DPB fullness is incremented by one and this picture is marked as "used for short-term reference".

When TwoVersionsOfCurrDecPicFlag is equal to 1, the current decoded picture before the invocation of the in-loop filter process as specified in clause F.8.7 is stored in the DPB in an empty picture storage buffer, the DPB fullness is incremented by one, and this picture is marked as "used for long-term reference".

NOTE – Unless more memory than required by the level limit is available for storage of decoded pictures, decoders should start storing decoded parts of the current picture into the DPB when the first slice segment is decoded and continue storing more decoded samples as the decoding process proceeds.

F.13.3.5 Removal of pictures from the DPB after decoding of the current picture

When TwoVersionsOfCurrDecPicFlag is equal to 1, immediately after decoding of the current picture, at the CPB removal time of the last decoding unit of access unit n (containing the current picture), the current decoded picture that is marked as "used for long-term reference" is removed from the DPB, and the DPB fullness is decremented by one.

F.13.4 Bitstream conformance

A bitstream of coded data conforming to this Specification shall fulfil all requirements specified in this clause.

The bitstream shall be constructed according to the syntax, semantics and constraints specified in this Specification outside of this annex.

The first access unit in a bitstream shall be an IRAP access unit.

When vps_base_layer_internal_flag is equal to 0, all the following bitstream conformance constraints apply only to coded pictures present in the bitstream and do not apply to pictures with nuh_layer_id equal to 0 which are provided by external means.

Let currPicLayerId be equal to the nuh_layer_id of the current picture.

For each current picture, let the variables maxPicOrderCnt and minPicOrderCnt be set equal to the maximum and the minimum, respectively, of the PicOrderCntVal values of the following pictures with nuh_layer_id equal to currPicLayerId:

- The current picture.
- The previous picture in decoding order that has TemporalId equal to 0 and that is not a RASL, RADL or SLNR picture.
- The short-term reference pictures in the RPS of the current picture.
- All pictures m that have PicOutputFlag equal to 1, AuCpbRemovalTime[m] less than AuCpbRemovalTime[currAu] and DpbOutputTime[m] greater than or equal to AuCpbRemovalTime[currAu], where currAu is the access unit containing the current picture, and AuCpbRemovalTime[m] and DpbOutputTime[m] are the CPB removal time and the DPB output time, respectively, of the access unit containing picture m.

The bitstream and the sub-bitstream of each output layer set are tested by the HRD for conformance as specified in clause C.1 and clause F.13.1. All the conditions specified in clause C.4 shall be fulfilled for each of the first and second sets of conformance tests. All of the following conditions shall be fulfilled for each of the third set of conformance tests:

1. For each partition unit n, with n greater than 0, associated with a buffering period SEI message, let the variable deltaTime90k[n] be specified as follows:

$$\text{deltaTime90k}[n] = 90\,000 * (\text{AuNominalRemovalTime}[n] - \text{AuFinalArrivalTime}[n-1]) \quad (\text{F-84})$$

The value of InitCpbRemovalDelay[SchedSelIdx] is constrained as follows:

- If cbr_flag[SchedSelIdx] is equal to 0, the following condition shall be true:

$$\text{InitCpbRemovalDelay}[\text{SchedSelIdx}] \leq \text{Ceil}(\text{deltaTime90k}[n]) \quad (\text{F-85})$$

- Otherwise (cbr_flag[SchedSelIdx] is equal to 1), the following condition shall be true:

$$\begin{aligned} \text{Floor}(\text{deltaTime90k}[n]) &\leq \text{InitCpbRemovalDelay}[\text{SchedSelIdx}] \\ &\leq \text{Ceil}(\text{deltaTime90k}[n]) \end{aligned} \quad (\text{F-86})$$

NOTE 1 – The exact number of bits in the BPB at the removal time of each partition unit may depend on which buffering period SEI message is selected to initialize the HRD. Encoders must take this into account to ensure that all

specified constraints must be obeyed regardless of which buffering period SEI message is selected to initialize the HRD, as the HRD may be initialized at any one of the buffering period SEI messages.

2. A BPB overflow is specified as the condition in which the total number of bits in the BPB is greater than the BPB size. The BPB shall never overflow.
3. When `low_delay_hrd_flag[HighestTid]` is equal to 0, the BPB shall never underflow. A BPB underflow is specified as follows:
 - If `SubHrdFlag` is equal to 0, a BPB underflow is specified as the condition in which the nominal BPB removal time of partition unit `n` `AuNominalRemovalTime[n]` is less than the final BPB arrival time of partition unit `n` `AuFinalArrivalTime[N]` for at least one value of `n`.
 - Otherwise (`SubHrdFlag` is equal to 1), a BPB underflow is specified as the condition in which the nominal BPB removal time of decoding unit `m` `DuNominalRemovalTime[m]` is less than the final BPB arrival time of decoding unit `m` `DuFinalArrivalTime[m]` for at least one value of `m`.
4. When `SubPicHrdFlag` is equal to 1, `low_delay_hrd_flag[HighestTid]` is equal to 1 and the nominal removal time of a decoding unit `m` of partition unit `n` is less than the final BPB arrival time of decoding unit `m` (i.e., `DuNominalRemovalTime[m] < DuFinalArrivalTime[m]`), the nominal removal time of partition unit `n` shall be less than the final BPB arrival time of partition unit `n` (i.e., `AuNominalRemovalTime[n] < AuFinalArrivalTime[n]`).
5. The nominal removal times of partition units from the BPB (starting from the second partition unit in decoding order) shall satisfy the constraints on `AuNominalRemovalTime[n]` and `AuCpbRemovalTime[n]` expressed in clauses A.4.1 through A.4.2 for bitstreams or layers conforming to profiles defined in Annex A, or expressed in clauses G.11.2.1 through G.11.2.2 for bitstreams or layers conforming to profiles defined in Annex G, or expressed in clauses H.11.2.1 through H.11.2.2 for bitstreams or layers conforming to profiles defined in Annex H.
6. For each current picture, after invocation of the process for removal of pictures from the sub-DPB as specified in clause F.13.3.2, the number of decoded pictures in the sub-DPB for the current layer, including all pictures `n`, where each picture `n` is contained in partition unit `kn`, in the current layer that are marked as "used for reference", or that have `PicOutputFlag` equal to 1 and `AuCpbRemovalTime[kn]` less than `AuCpbRemovalTime[currAu]`, where `currAu` is the partition unit containing the current picture, shall be less than or equal to `max_vps_dec_pic_buffering_minus1[TargetOlsIdx][layerIdx][HighestTid]`, where `layerIdx` is equal to the value such that `LayerSetLayerIdList[TargetDecLayerSetIdx][layerIdx]` is equal to `currPicLayerId`.
7. All reference pictures shall be present in the DPB when needed for prediction. Each picture that has `PicOutputFlag` equal to 1 shall be present in the DPB at its DPB output time unless it is removed from the DPB before its output time by one of the processes specified in clause F.13.3.
8. For each current picture that is not an IRAP picture with `NoRasOutputFlag` equal to 1 or that is not the first picture of the current layer with `nuh_layer_id` greater than 0 that follows an IRAP picture that has `nuh_layer_id` equal to 0 and has `NoClasOutputFlag` equal to 1, the value of `maxPicOrderCnt – minPicOrderCnt` shall be less than `MaxPicOrderCntLsb / 2`.
9. The value of `DpbOutputInterval[n]` as given by Equation F-83, which is the difference between the output time of a partition unit that has `AuOutputFlag` equal to 1 and that of the first partition unit following it in output order and having `AuOutputFlag` equal to 1, shall satisfy the constraint on `DpbOutputInterval[n]` expressed in clause A.4.2 for the profile, tier and level of the bitstream or layer using the decoding process specified in clauses 2 through 10, or expressed in clause G.11.2.2 for the profile, tier and level of the bitstream or layer using the decoding process specified in clauses F.2 through F.10 and either clauses G.2 through G.10 or clauses H.2 through H.10.
10. For each current picture, when `sub_pic_cpb_params_in_pic_timing_sei_flag` is equal to 1, let `tmpCpbRemovalDelaySum` be derived as follows:

$$\begin{aligned}
 &\text{tmpCpbRemovalDelaySum} = 0 \\
 &\text{for}(i = 0; i < \text{num_decoding_units_minus1}; i++) \\
 &\quad \text{tmpCpbRemovalDelaySum} += \text{du_cpb_removal_delay_increment_minus1}[i] + 1
 \end{aligned}
 \tag{F-87}$$

The value of `ClockSubTick * tmpCpbRemovalDelaySum` shall be equal to the difference between the nominal BPB removal time of the current partition unit and the nominal BPB removal time of the first decoding unit in the current partition unit in decoding order.

11. Let `picA` be an IRAP picture with `NoRasOutputFlag` equal to 1 and belonging to a layer `layerA`. Let `auB` be the earlier, in decoding order, of the first partition unit containing an IRAP picture with `nuh_layer_id` equal to 0 and `NoClasOutputFlag` equal to 1 that succeeds `picA` in decoding order and the first partition unit containing an IRAP picture with `NoRasOutputFlag` equal to 1 in `layerA` that succeeds `picA` in decoding order. For any two pictures `picM` and `picN` in the layer `layerA` contained in partition units `m` and `n`, respectively, that either are `picA` or succeed

picA in decoding order and precede auB in decoding order, when DpbOutputTime[m] is greater than DpbOutputTime[n], PicOrderCnt(picM) shall be greater than PicOrderCnt(picN), where PicOrderCnt(picM) and PicOrderCnt(picN) are the PicOrderCntVal values of picM and picN, respectively, immediately after the invocation of the decoding process for picture order count of the latter of picM and picN in decoding order.

NOTE 2 – All pictures of an earlier CVS in decoding order that are output are output before any pictures of a later CVS in decoding order. Within any particular CVS where all pictures have poc_reset_id not present or equal to 0, the pictures that are output are output in increasing PicOrderCntVal order. For any particular CVS where some pictures have poc_reset_id greater than 0, within each POC resetting period, the pictures that are output are output in increasing PicOrderCntVal order, and for any two pictures that are output and are in different POC resetting periods, the one that is earlier in decoding order is output earlier.

12. For any decoding unit m, DuCpbRemovalTime[m] shall be greater than or equal to the BPB removal time of the previous DU that precedes the current DU in decoding order and that is in the same bitstream partition as the decoding unit m or includes one or more VCL NAL units of a picture that may be used as a direct or indirect reference picture for the picture containing the one or more VCL NAL units included in the decoding unit m.
13. The DPB output times derived for all pictures in any particular access unit shall be the same.

F.13.5 Decoder conformance

F.13.5.1 General

A decoder conforming to this Specification shall fulfil all requirements specified in this clause.

A decoder claiming conformance to a specific profile, tier and level shall be able to successfully decode all bitstreams that conform to the bitstream conformance requirements specified in clause F.13.4, in the manner specified in Annexes A, G and H for profile, tier and level specified in Annexes A, G and H, respectively, provided that all VPSs, SPSs and PPSs referred to by the VCL NAL units, appropriate buffering period, picture timing and decoding unit information SEI messages are conveyed to the decoder, in a timely manner, either in the bitstream (by non-VCL NAL units), or by external means not specified in this Specification, and, when vps_base_layer_internal_flag is equal to 0, the decoded pictures with nuh_layer_id equal to 0 and their properties as specified in clause F.8.1 and its subclauses are conveyed to the decoder in a timely manner by external means not specified in this Specification.

When a bitstream contains syntax elements that have values that are specified as reserved and it is specified that decoders shall ignore values of the syntax elements or NAL units containing the syntax elements having the reserved values, and the bitstream is otherwise conforming to this Specification, a conforming decoder shall decode the bitstream in the same manner as it would decode a conforming bitstream and shall ignore the syntax elements or the NAL units containing the syntax elements having the reserved values as specified.

There are two types of conformance that can be claimed by a decoder: output timing conformance and output order conformance.

To check conformance of a decoder, test bitstreams containing one or more bitstream partitions conforming to the claimed profile, tier and level, as specified in clause F.13.4 are delivered by a hypothetical stream scheduler (HSS) both to the HRD and to the decoder under test (DUT). When vps_base_layer_internal_flag is equal to 0, decoded pictures with nuh_layer_id equal to 0 and their properties as specified in clause F.8.1 and its subclauses are also conveyed both to the HRD and to the DUT in a timely manner by external means not specified in this Specification. All cropped decoded pictures output by the HRD shall also be output by the DUT, each cropped decoded picture output by the DUT shall be a picture with PicOutputFlag equal to 1, and, for each such cropped decoded picture output by the DUT, the values of all samples that are output shall be equal to the values of the samples produced by the specified decoding process. The flag BaseLayerOutputFlag and all flags BaseLayerPicOutputFlag output by the HRD shall also be output by the DUT, and the values that are output shall be equal to the values produced by the specified decoding process.

For output timing decoder conformance, the HSS operates as described above, with delivery schedules selected only from the subset of values of SchedSelIdx for which the bit rate and CPB size are restricted as specified in Annexes A, G and H for the specified profile, tier and level in Annexes A, G and H, respectively, or with "interpolated" delivery schedules as specified below for which the bit rate and CPB size are restricted as specified in Annexes A, G and H for the specified profile, tier and level in Annexes A, G and H, respectively. The same delivery schedule is used for both the HRD and the DUT.

When the HRD parameters and the buffering period SEI messages are present with the number of signalled delivery schedules greater than 0, the decoder shall be capable of decoding each bitstream partition as delivered from the HSS operating using an "interpolated" delivery schedule specified as having peak bit rate r, CPB size c(r) and initial CPB removal delay (f(r) ÷ r) as follows:

$$\alpha = (r - \text{BitRate}[\text{SchedSelIdx} - 1]) \div (\text{BitRate}[\text{SchedSelIdx}] - \text{BitRate}[\text{SchedSelIdx} - 1]),$$

(F-88)

$$c(r) = \alpha * CpbSize[SchedSelIdx] + (1 - \alpha) * CpbSize[SchedSelIdx - 1], \quad (F-89)$$

$$f(r) = \alpha * InitCpbRemovalDelay[SchedSelIdx] * BitRate[SchedSelIdx] + (1 - \alpha) * InitCpbRemovalDelay[SchedSelIdx - 1] * BitRate[SchedSelIdx - 1] \quad (F-90)$$

for any SchedSelIdx > 0 and r such that BitRate[SchedSelIdx - 1] <= r <= BitRate[SchedSelIdx] such that r and c(r) are within the limits as specified in Annexes A, G and H for the maximum bit rate and buffer size for the specified profile, tier and level defined in Annexes A, G and H, respectively. The following applies for the specifications above:

- Let combIdx be any value in the range of 0 to num_bsp_schedules_minus1[TargetOlsIdx][TargetPsIdx][HighestTid], inclusive.
- SchedSelIdx shall be one of the values among bsp_sched_idx[TargetOlsIdx][TargetPsIdx][HighestTid][combIdx][j] for and j is the index of the bitstream partition index of TargetBitstreamPartition.
- SchedSelIdx - 1 is to be understood as bsp_sched_idx[TargetOlsIdx][TargetPsIdx][HighestTid][combIdx - 1][j] for any value of k in the range of 1 to combIdx, inclusive.

NOTE 1 – InitCpbRemovalDelay[SchedSelIdx] can be different from one buffering period to another and need to be recalculated.

For output timing decoder conformance, an HRD as described above is used and the timing (relative to the delivery time of the first bit) of picture output is the same for both the HRD and the DUT up to a fixed delay.

For output order decoder conformance, the following applies for each bitstream partition in TargetPartitioningScheme:

- The HSS delivers the TargetBitstreamPartition to the DUT "by demand" from the DUT, meaning that the HSS delivers bits (in decoding order) only when the DUT requires more bits to proceed with its processing.
NOTE 2 – This means that for this test, the coded picture buffer of the DUT could be as small as the size of the largest decoding unit.
- A modified HRD as described below is used, and the HSS delivers the bitstream to the HRD by one of the schedules specified in the TargetBitstreamPartition such that the bit rate and CPB size are restricted as specified in Annex A for profiles defined in Annex A, Annex G for profiles defined in Annex G and Annex H for profiles defined in Annex H. The order of pictures output shall be the same for both the HRD and the DUT.
- The HRD BPB size is given by CpbSize[SchedSelIdx] as specified in clause F.13.1. The sub-DPB size of the sub-DPB for a layer with nuh_layer_id equal to currLayerId is max_vps_dec_pic_buffering_minus1[TargetOlsIdx][layerIdx][HighestTid] + 1, where layerIdx is equal to the value such that LayerSetLayerIdList[TargetDecLayerSetIdx][layerIdx] is equal to currLayerId. The removal time from the CPB for the HRD is the final bit arrival time and decoding is immediate. The operation of the DPB of this HRD is as described in the subclauses of clause F.13.5.2.

F.13.5.2 Operation of the output order DPB

F.13.5.2.1 General

The decoded picture buffer consists of sub-DPBs and each sub-DPB contains picture storage buffers for storage of decoded pictures of one layer. Each of the picture storage buffers of a sub-DPB contains a decoded picture that is marked as "used for reference" or is held for future output.

The process for output and removal of pictures from the DPB before decoding of the current picture as specified in clause F.13.5.2.2 is invoked, followed by the invocation of the process for current decoded picture marking and storage as specified in clause F.13.3.4, further followed by the invocation of the process for removal of pictures from the DPB after decoding of the current picture as specified in clause F.13.3.5, and finally followed by the invocation of the process for additional bumping as specified in clause F.13.5.2.3. The "bumping" process is specified in clause F.13.5.2.4 and is invoked as specified in clauses F.13.5.2.2 and F.13.5.2.3.

These processes are applied independently for each layer, starting from the base layer, in increasing order of the nuh_layer_id values of the layers in the bitstream. When these processes are applied for a particular layer, only the sub-DPB for the particular layer is affected except for the "bumping" process, which may crop and output pictures, mark pictures as "not needed for output" and empty picture storage buffers for any layer.

NOTE – In the operation of output order DPB, same as in the operation of output timing DPB, decoded pictures with PicOutputFlag equal to 1 in the same access unit are also output consecutively in ascending order of the nuh_layer_id values of the decoded pictures.

Let picture n and the current picture be the coded picture or decoded picture of the access unit n for a particular value of nuh_layer_id, wherein n is a non-negative integer number.

When these processes are applied for a layer with nuh_layer_id equal to currLayerId, the variables MaxNumReorderPics, MaxLatencyIncreasePlus1, MaxLatencyValue and MaxDecPicBufferingMinus1 are derived as follows:

- MaxNumReorderPics is set equal to max_vps_num_reorder_pics[TargetOlsIdx][HighestTid] of the active VPS.

- MaxLatencyIncreasePlus1 is set equal to the value of the syntax element `max_vps_latency_increase_plus1[TargetOlsIdx][HighestTid]` of the active VPS.
- MaxLatencyValue is set equal to `MaxVpsLatencyPictures[TargetOlsIdx][HighestTid]` of the active VPS.
- MaxDecPicBufferingMinus1 is set equal to the value of the syntax element `max_vps_dec_pic_buffering_minus1[TargetOlsIdx][layerIdx][HighestTid]` of the active VPS, where `layerIdx` is equal to the value such that `LayerSetLayerIdList[TargetDecLayerSetIdx][layerIdx]` is equal to `currLayerId`.

F.13.5.2.2 Output and removal of pictures from the DPB before decoding of the current picture

When the current picture is not picture 0 in the current layer, the output and removal of pictures in the current layer, with `nuh_layer_id` equal to `currLayerId`, from the DPB before the decoding of the current picture, i.e., picture `n`, but after parsing the slice header of the first slice of the current picture and before the invocation of the decoding process for picture order count, happens instantaneously when the first decoding unit of the current picture is removed from the CPB and proceeds as follows:

- When the current picture is a POC resetting picture, all pictures in the DPB that do not belong to the current access unit and that are marked as "needed for output" are output, starting with pictures with the smallest value of `PicOrderCntVal` of all pictures excluding those in the current access unit in the DPB, in ascending order of the `PicOrderCntVal` values, and pictures with the same value of `PicOrderCntVal` are output in ascending order of the `nuh_layer_id` values. When a picture is output, it is cropped using the conformance cropping window specified in the active SPS for the picture, the cropped picture is output, and the picture is marked as "not needed for output".
- The decoding processes for picture order count and RPS are invoked. When decoding a CVS conforming to one or more of the profiles specified in Annex A using the decoding process specified in clauses 2 through 10, the decoding processes for picture order count and RPS that are invoked are as specified in clauses 8.3.1 and 8.3.2, respectively. When decoding a CVS conforming to one or more of the profiles specified in Annex G or H using the decoding process specified in Annex F, and Annex G or H, the decoding processes for picture order count and RPS that are invoked are as specified in clauses F.8.3.1 and F.8.3.2, respectively.
- The variable `listOfSubDpbsToEmpty` is derived as follows:
 - If a new VPS is activated by the current access unit or the current picture is IRAP picture with `nuh_layer_id` equal to `SmallestLayerId`, `NoRasOutputFlag` equal to 1 and `NoClasOutputFlag` equal to 1, `listOfSubDpbsToEmpty` is set equal to all the sub-DPBs.
 - Otherwise, if the current picture is an IRAP picture with any `nuh_layer_id` value `indepLayerId` such that `NumDirectRefLayers[indepLayerId]` is equal to 0 and `indepLayerId` is greater than `SmallestLayerId`, and with `NoRasOutputFlag` equal to 1, and `LayerResetFlag` is equal to 1, `listOfSubDpbsToEmpty` is set equal to the sub-DPBs containing the current layer and the sub-DPBs containing the predicted layers of the current layer.
 - Otherwise, `listOfSubDpbsToEmpty` is set equal to the sub-DPB containing the current layer.
- If the current picture is an IRAP picture with `NoRasOutputFlag` equal to 1 and any of the following conditions is true:
 - `nuh_layer_id` equal to `SmallestLayerId`,
 - `nuh_layer_id` of the current layer is greater than `SmallestLayerId` and `NumDirectRefLayers[nuh_layer_id]` is equal to 0,

the following ordered steps are applied:

1. The variable `NoOutputOfPriorPicsFlag` is derived for the decoder under test as follows:
 - If the current picture is a CRA picture, `NoOutputOfPriorPicsFlag` is set equal to 1 (regardless of the value of `no_output_of_prior_pics_flag`).
 - Otherwise, if the value of `pic_width_in_luma_samples`, `pic_height_in_luma_samples`, `chroma_format_idc`, `bit_depth_luma_minus8`, `bit_depth_chroma_minus8`, `separate_colour_plane_flag` or `sps_max_dec_pic_buffering_minus1[HighestTid]` derived from the active SPS for the current layer is different from the value of `pic_width_in_luma_samples`, `pic_height_in_luma_samples`, `chroma_format_idc`, `bit_depth_luma_minus8`, `bit_depth_chroma_minus8`, `separate_colour_plane_flag` or `sps_max_dec_pic_buffering_minus1[HighestTid]`, respectively, derived from the SPS that was active for the current layer when decoding the preceding picture in the current layer, `NoOutputOfPriorPicsFlag` may (but should not) be set equal to 1 by the decoder under test, regardless of the value of `no_output_of_prior_pics_flag`.

NOTE – Although setting `NoOutputOfPriorPicsFlag` equal to `no_output_of_prior_pics_flag` is preferred under these conditions, the decoder under test is allowed to set `NoOutputOfPriorPicsFlag` to 1 in this case.

- Otherwise, NoOutputOfPriorPicsFlag is set equal to no_output_of_prior_pics_flag.
- 2. The value of NoOutputOfPriorPicsFlag derived for the decoder under test is applied for the HRD as follows:
 - If NoOutputOfPriorPicsFlag is equal to 0, all non-empty picture storage buffers in all the sub-DPBs included in listOfSubDpbsToEmpty are output by repeatedly invoking the "bumping" process specified in clause F.13.5.2.4 until all these pictures are marked as "not needed for output".
 - Otherwise (NoOutputOfPriorPicsFlag is equal to 1), all picture storage buffers containing a picture that is marked as "not needed for output" and "unused for reference" are emptied (without output), all pictures that are contained in a sub-DPB included in listOfSubDpbsToEmpty are emptied, and the sub-DPB fullness of each sub-DPB is decremented by the number of picture storage buffers emptied in that sub-DPB.
- Otherwise, all picture storage buffers that contain a picture in the current layer and that are marked as "not needed for output" and "unused for reference" are emptied (without output). For each picture storage buffer that is emptied, the sub-DPB fullness is decremented by one. When one or more of the following conditions are true, the "bumping" process specified in clause F.13.5.2.4 is invoked repeatedly until none of the following conditions are true:
 - The number of access units that contain at least one decoded picture in the DPB marked as "needed for output" is greater than MaxNumReorderPics.
 - MaxLatencyIncreasePlus1 is not equal to 0 and there is at least one access unit that contains at least one decoded picture in the DPB marked as "needed for output" for which the associated variable PicLatencyCount is greater than or equal to MaxLatencyValue.
 - The number of pictures in the sub-DPB is greater than or equal to MaxDecPicBufferingMinus1 + 1 – TwoVersionsOfCurrDecPicFlag.

F.13.5.2.3 Additional bumping

The processes specified in this clause happen instantaneously when the last decoding unit of picture *n* is removed from the CPB.

When the current picture is the last picture in an access unit, the following applies for each decoded picture with *nuh_layer_id* greater than or equal to (*vps_base_layer_internal_flag* ? 0 : 1) of the access unit:

- If the decoded picture has PicOutputFlag equal to 1, it is marked as "needed for output".
- Otherwise (the decoded picture has PicOutputFlag equal to 0), it is marked as "not needed for output".

NOTE – Prior to investigating the conditions above, PicOutputFlag of each picture of the access unit is updated as specified in clause F.8.1.2.

When one or more of the following conditions are true, the "bumping" process specified in clause F.13.5.2.4 is invoked repeatedly until none of the following conditions are true:

- The number of access units that contain at least one decoded picture in the DPB marked as "needed for output" is greater than MaxNumReorderPics.
- MaxLatencyIncreasePlus1 is not equal to 0 and there is at least one access unit that contains at least one decoded picture in the DPB marked as "needed for output" for which the associated variable PicLatencyCount is greater than or equal to MaxLatencyValue.

F.13.5.2.4 "Bumping" process

The "bumping" process consists of the following ordered steps:

1. The picture or pictures that are first for output are selected as the ones having the smallest value of PicOrderCntVal of all pictures in the DPB marked as "needed for output".
2. Each of these pictures is, in ascending *nuh_layer_id* order, cropped, using the conformance cropping window specified in the active SPS for the picture, the cropped picture is output and the picture is marked as "not needed for output".
3. Each picture storage buffer that contains a picture marked as "unused for reference" and that was one of the pictures cropped and output is emptied and the fullness of the associated sub-DPB is decremented by one.

F.13.6 Demultiplexing process for deriving a bitstream partition

Inputs to this process are a bitstream, a layer identifier list *bspLayerId*[*bspIdx*] and the number of layer identifiers *numBspLayers* in the layer index list *bspLayerId*[*bspIdx*].

Output of this process is a bitstream partition.

Let the variable *minBspLayerId* be the smallest value of *bspLayerId*[*bspIdx*] with any value of *bspIdx* in the range of 0 to *numBspLayers* – 1, inclusive.

The output bitstream partition consists of selected NAL units of the input bitstream in the same order as they appear in the input bitstream. The following NAL units of the input bitstream are omitted from the output bitstream partition, while the remaining NAL units of the input bitstream are included in the output bitstream partition:

- Omit all NAL units that have a `nuh_layer_id` value other than `bspLayerId[bspIdx]` with any value of `bspIdx` in the range of 0 to `numBspLayers – 1`, inclusive.
- Omit all SEI NAL units containing a scalable nesting SEI message for which no derived `nestingLayerIdList[i]` contains any layer identifier value equal to `bspLayerId[bspIdx]` with any value of `bspIdx` in the range of 0 to `numBspLayers – 1`, inclusive.
- Omit all SEI NAL units containing a scalable nesting SEI message for which a derived `nestingLayerIdList[i]` contains a layer identifier value less than `minBspLayerId`.

F.14 Supplemental enhancement information

F.14.1 General

The specifications in clause D.1 apply.

F.14.2 SEI payload syntax

F.14.2.1 General SEI payload syntax

The specifications in clause D.2.1 apply.

F.14.2.2 Annex D SEI message syntax for multi-layer extensions

The specifications in clauses D.2.2 through D.2.49 apply.

F.14.2.3 Layers not present SEI message syntax

<code>layers_not_present(payloadSize) {</code>	Descriptor
<code>lnp_sei_active_vps_id</code>	<code>u(4)</code>
<code>for(i = 0; i <= MaxLayersMinus1; i++)</code>	
<code>layer_not_present_flag[i]</code>	<code>u(1)</code>
<code>}</code>	

F.14.2.4 Inter-layer constrained tile sets SEI message syntax

	Descriptor
inter_layer_constrained_tile_sets (payloadSize) {	
il_all_tiles_exact_sample_value_match_flag	u(1)
il_one_tile_per_tile_set_flag	u(1)
if(!il_one_tile_per_tile_set_flag) {	
il_num_sets_in_message_minus1	ue(v)
if(il_num_sets_in_message_minus1)	
skipped_tile_set_present_flag	u(1)
numSignificantSets = il_num_sets_in_message_minus1 – skipped_tile_set_present_flag + 1	
for(i = 0; i < numSignificantSets; i++) {	
ilcts_id [i]	ue(v)
il_num_tile_rects_in_set_minus1 [i]	ue(v)
for(j = 0; j <= il_num_tile_rects_in_set_minus1[i]; j++) {	
il_top_left_tile_idx [i][j]	ue(v)
il_bottom_right_tile_idx [i][j]	ue(v)
}	
ilc_idc [i]	u(2)
if(!il_all_tiles_exact_sample_value_match_flag)	
il_exact_sample_value_match_flag [i]	u(1)
}	
} else	
all_tiles_ilc_idc	u(2)
}	

F.14.2.5 Bitstream partition nesting SEI message syntax

	Descriptor
bsp_nesting (payloadSize) {	
sei_ols_idx	ue(v)
sei_partitioning_scheme_idx	ue(v)
bsp_idx	ue(v)
num_seis_in_bsp_minus1	ue(v)
while(!byte_aligned())	
bsp_nesting_zero_bit /* equal to 0 */	u(1)
for(i = 0; i <= num_seis_in_bsp_minus1; i++)	
sei_message()	
}	

F.14.2.6 Bitstream partition initial arrival time SEI message syntax

	Descriptor
bsp_initial_arrival_time(payloadSize) {	
psIdx = sei_partitioning_scheme_idx	
if(nalInitialArrivalDelayPresent)	
for(i = 0; i < BspSchedCnt[sei_ols_idx][psIdx][MaxTemporalId[0]]; i++)	
nal_initial_arrival_delay[i]	u(v)
if(vclInitialArrivalDelayPresent)	
for(i = 0; i < BspSchedCnt[sei_ols_idx][psIdx][MaxTemporalId[0]]; i++)	
vcl_initial_arrival_delay[i]	u(v)
}	

F.14.2.7 Sub-bitstream property SEI message syntax

	Descriptor
sub_bitstream_property(payloadSize) {	
sb_property_active_vps_id	u(4)
num_additional_sub_streams_minus1	ue(v)
for(i = 0; i <= num_additional_sub_streams_minus1; i++) {	
sub_bitstream_mode[i]	u(2)
ols_idx_to_vps[i]	ue(v)
highest_sublayer_id[i]	u(3)
avg_sb_property_bit_rate[i]	u(16)
max_sb_property_bit_rate[i]	u(16)
}	
}	

F.14.2.8 Alpha channel information SEI message syntax

	Descriptor
alpha_channel_info(payloadSize) {	
alpha_channel_cancel_flag	u(1)
if(!alpha_channel_cancel_flag) {	
alpha_channel_use_idc	u(3)
alpha_channel_bit_depth_minus8	u(3)
alpha_transparent_value	u(v)
alpha_opaque_value	u(v)
alpha_channel_incr_flag	u(1)
alpha_channel_clip_flag	u(1)
if(alpha_channel_clip_flag)	
alpha_channel_clip_type_flag	u(1)
}	
}	

F.14.2.9 Overlay information SEI message syntax

overlay_info(payloadSize) {	Descriptor
overlay_info_cancel_flag	u(1)
if(!overlay_info_cancel_flag) {	
overlay_content_aux_id_minus128	ue(v)
overlay_label_aux_id_minus128	ue(v)
overlay_alpha_aux_id_minus128	ue(v)
overlay_element_label_value_length_minus8	ue(v)
num_overlays_minus1	ue(v)
for(i = 0; i <= num_overlays_minus1; i++) {	
overlay_idx[i]	ue(v)
language_overlay_present_flag[i]	u(1)
overlay_content_layer_id[i]	u(6)
overlay_label_present_flag[i]	u(1)
if(overlay_label_present_flag[i])	
overlay_label_layer_id[i]	u(6)
overlay_alpha_present_flag[i]	u(1)
if(overlay_alpha_present_flag[i])	
overlay_alpha_layer_id[i]	u(6)
if(overlay_label_present_flag[i]) {	
num_overlay_elements_minus1[i]	ue(v)
for(j = 0; j <= num_overlay_elements_minus1[i]; j++) {	
overlay_element_label_min[i][j]	u(v)
overlay_element_label_max[i][j]	u(v)
}	
}	
}	
while(!byte_aligned())	
overlay_zero_bit /* equal to 0 */	f(1)
for(i = 0; i <= num_overlays_minus1; i++) {	
if(language_overlay_present_flag[i])	
overlay_language[i]	st(v)
overlay_name[i]	st(v)
if(overlay_label_present_flag[i])	
for(j = 0; j <= num_overlay_elements_minus1[i]; j++)	
overlay_element_name[i][j]	st(v)
}	
overlay_info_persistence_flag	u(1)
}	
}	

F.14.2.10 Temporal motion vector prediction constraints SEI message syntax

temporal_mv_prediction_constraints(payloadSize) {	Descriptor
prev_pics_not_used_flag	u(1)
no_intra_layer_col_pic_flag	u(1)
}	

F.14.2.11 Frame-field information SEI message syntax

frame_field_info(payloadSize) {	Descriptor
ffinfo_pic_struct	u(4)
ffinfo_source_scan_type	u(2)
ffinfo_duplicate_flag	u(1)
}	

F.14.3 SEI payload semantics

F.14.3.1 General SEI payload semantics

The general SEI payload semantics specified in clause D.3.1 apply with the following modifications and additions:

The list VclAssociatedSeiList is set to consist of the payloadType values 2, 3, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 131, 132, 134 to 152, inclusive, 154 to 159, inclusive, 161, 165, 167, 168, 200 to 202, inclusive, and 205.

The list PicUnitRepConSeiList is set to consist of the payloadType values 0, 1, 2, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 129, 131, 132, 133, 135 to 152, inclusive, 154 to 168, inclusive, 200 to 202, inclusive, and 205.

The semantics and persistence scope for each SEI message specified in this annex are specified in the semantics specification for each particular SEI message in clauses F.14.3.3 to F.14.3.11, inclusive.

NOTE – Persistence information for SEI messages specified in this annex is informatively summarized in Table F.4.

Table F.4 – Persistence scope of SEI messages (informative)

SEI message	Persistence scope
Layers not present	Specified by the semantics of the SEI message
Inter-layer constrained tile sets	The CLVS associated with the SEI message
Bitstream partition nesting	Each SEI message contained in the bitstream partition nesting SEI message has the same persistence scope as if the SEI message was not contained in the bitstream partition nesting SEI message
Bitstream partition initial arrival time	The remainder of the bitstream partition (specified by the containing bitstream partition nesting SEI message)
Sub-bitstream property	The CVS containing the SEI message
Alpha channel information	Specified by the syntax of the SEI message
Overlay information	Specified by the syntax of the SEI message
Temporal motion vector prediction constraints	Specified by the semantics of the SEI message
Frame-field information	The access unit containing the SEI message

Let prevVclNalUnitInAu of an SEI NAL unit or an SEI message be the preceding VCL NAL unit in decoding order, if any, in the same access unit, and nextVclNalUnitInAu of an SEI NAL unit or an SEI message be the next VCL NAL unit in decoding order, if any, in the same access unit. It is a requirement of bitstream conformance that the following restrictions apply:

- When a buffering period SEI message, a picture timing SEI message, a decoding unit information SEI message or a bitstream partition initial arrival time SEI message is present in a bitstream partition nesting SEI message contained in a scalable nesting SEI message, the scalable nesting SEI message shall not follow any other SEI message that

follows the prevVclNalUnitInAu of the scalable nesting SEI message and precedes the nextVclNalUnitInAu of the scalable nesting SEI message, other than an active parameter sets SEI message, a non-scalable-nested buffering period SEI message, a non-scalable-nested picture timing SEI message, a non-scalable-nested decoding unit information SEI message, a scalable nesting SEI message including a buffering period SEI message, a picture timing SEI message, a decoding unit information SEI message or another scalable nesting SEI message that contains a bitstream partition nesting SEI message including a buffering period SEI message, a picture timing SEI message, a decoding unit information SEI message or a bitstream partition initial arrival time SEI message.

F.14.3.2 Annex D SEI message semantics for multi-layer extensions

F.14.3.2.1 General

The semantics of SEI messages specified in clauses D.3.2 through D.3.49 apply with the modifications specified in clauses F.14.3.2.2 through F.14.3.2.8.

F.14.3.2.2 Buffering period SEI message semantics for multi-layer extensions

The specifications of clause D.3.2 apply with the following modifications and additions:

When the buffering period SEI message is directly contained in a scalable nesting SEI message within an SEI NAL unit with a nuh_layer_id value targetNuhLayerId greater than 0, the semantics of clause D.3.2 apply to the output bitstream of the independent non-base layer rewriting process of clause F.10.2 with the input variables assignedBaseLayerId equal to nestingLayerIdList[i][0] and tldTarget equal to MaxTemporalId[i] for each value of i in the range of 0 to nestingNumOps – 1, inclusive.

When the buffering period SEI message is contained in a bitstream partition nesting SEI message, the following applies:

- A set of skipped leading pictures skippedPictureList consists of the CL-RAS pictures and the RASL pictures associated with the IRAP pictures with nuh_layer_id equal to nuhLayerId for which LayerInitializedFlag[nuhLayerId] is equal to 0 at the start of decoding the IRAP picture and for which nuhLayerId is among the nestingLayerIdList[i] for any value of i in the range of 0 to nesting_num_ops_minus1, inclusive.
- A picture is said to be a notDiscardablePic picture when the picture has TemporalId equal to 0, has discardable_flag equal to 0, and is not a RASL, RADL or SLNR picture.
- The variables olsIdx, psIdx and bspIdx are set equal to sei_ols_idx, sei_partitioning_scheme_idx and bsp_idx, respectively, of the bitstream partition nesting SEI message containing this buffering period SEI message and the bspIdx-th bitstream partition of the psIdx-th partitioning scheme of the olsIdx-th OLS is referred to as the current bitstream partition.
- The variable maxTId is set equal to MaxTemporalId[0] for the scalable nesting SEI message containing the bitstream partition nesting SEI message containing this buffering period SEI message.
- The variable CpbCnt is set equal to BspSchedCnt[olsIdx][psIdx][maxTId] for the syntax and semantics of the buffering period SEI message and for the constraints specified below.
- The variable hrdParamIdx[i] is set equal to the value of bsp_hrd_idx[olsIdx][psIdx][maxTId][i][bspIdx] for each value of i in the range of 0 to CpbCnt – 1, inclusive.
- The syntax elements au_cpb_removal_delay_length_minus1, dpb_output_delay_length_minus1 and sub_pic_hrd_params_present_flag are found in or derived from the hrdParamIdx[i]-th hrd_parameters() syntax structure for any value of i in the range of 0 to CpbCnt – 1, inclusive. It is a requirement of bitstream conformance that the values of au_cpb_removal_delay_length_minus1, dpb_output_delay_length_minus1 and sub_pic_hrd_params_present_flag derived using a particular value of i shall be the same as the value of the respective syntax elements derived using any other allowed value of i.
- The syntax element initial_cpb_removal_delay_length_minus1 is found in or derived from the hrdParamIdx[i]-th hrd_parameters() syntax structure for the semantics of nal_initial_cpb_removal_delay[i], nal_initial_alt_cpb_removal_delay[i], nal_initial_cpb_removal_offset[i], nal_initial_alt_cpb_removal_offset[i], vcl_initial_cpb_removal_delay[i], vcl_initial_alt_cpb_removal_delay[i], vcl_initial_cpb_removal_offset[i] and vcl_initial_alt_cpb_removal_offset[i].
- The variable CpbSize[i] is set equal to BpbSize[olsIdx][psIdx][maxTId][i][bspIdx] for the semantics of nal_initial_cpb_removal_delay[i], nal_initial_alt_cpb_removal_delay[i], vcl_initial_cpb_removal_delay[i] and vcl_initial_alt_cpb_removal_delay[i].
- The variable BitRate[i] is set equal to BpBitRate[olsIdx][psIdx][maxTId][i][bspIdx] for the semantics of nal_initial_cpb_removal_delay[i], nal_initial_alt_cpb_removal_delay[i], vcl_initial_cpb_removal_delay[i] and vcl_initial_alt_cpb_removal_delay[i].

- The variables `NalHrdBpPresentFlag` and `VclHrdBpPresentFlag` are derived from the `hrdParamIdx[i]`-th `hrd_parameters()` syntax structure for any value of `i` in the range of 0 to `CpbCnt – 1`, inclusive. It is a requirement of bitstream conformance that the value of `NalHrdBpPresentFlag` shall be the same regardless of the value of `i` used in its derivation. It is a requirement of bitstream conformance that the value of `VclHrdBpPresentFlag` shall be the same regardless of the value of `i` used in its derivation.

The presence of buffering period SEI messages for the current bitstream partition is specified as follows on the basis of the variables `NalHrdBpPresentFlag` and `VclHrdBpPresentFlag` that are derived as specified above:

- If `NalHrdBpPresentFlag` is equal to 1 or `VclHrdBpPresentFlag` is equal to 1, the following applies for each access unit in the CVS:
 - If the access unit is an IRAP access unit, a buffering period SEI message applicable to the current bitstream partition shall be associated with the access unit.
 - Otherwise, if the access unit contains a `notDiscardablePic` in at least one layer included in the current bitstream partition, a buffering period SEI message applicable to the current bitstream partition may or may not be associated with the access unit.
 - Otherwise, the access unit shall not be associated with a buffering period SEI message applicable to the current bitstream partition.
- Otherwise (`NalHrdBpPresentFlag` and `VclHrdBpPresentFlag` are both equal to 0), no access unit in the CVS shall be associated with a buffering period SEI message applicable to the current bitstream partition.

When the buffering period SEI message is contained in a bitstream partition nesting SEI message, the following semantics of `concatenation_flag` and `au_cpb_removal_delay_delta_minus1` replace those specified in clause D.3.2 and the following specifications apply to each picture `currPic` with `nuh_layed_id` `currNuhLayerId` equal to any value included in the current bitstream partition in the current access unit.

When the picture `currPic` with `nuh_layer_id` equal to `targetLayerId` in the current access unit is not the first picture with that `nuh_layer_id` value in the bitstream in decoding order, let `prevNonDiscardablePic` be the preceding picture in decoding order with that `nuh_layer_id` value and with `TemporalId` equal to 0 that is not a RASL, RADL or SLNR picture.

concatenation_flag indicates, when the picture `currPic` is not the first picture with `nuh_layer_id` equal to `currNuhLayerId` in the bitstream in decoding order, whether the nominal CPB removal time of the current picture is determined relative to the nominal CPB removal time of the preceding picture with a buffering period SEI message that applies to the current bitstream partition, or relative to the nominal CPB removal time of the picture `prevNonDiscardablePic`.

au_cpb_removal_delay_delta_minus1 plus 1, when the picture `currPic` is not the first picture with `nuh_layer_id` equal to `currNuhLayerId` in the bitstream in decoding order, specifies a CPB removal delay increment value relative to the nominal CPB removal time of the picture `prevNonDiscardablePic`. This syntax element has a length in bits given by `au_cpb_removal_delay_length_minus1 + 1`.

When the buffering period SEI message is contained in a bitstream partition nesting SEI message and `concatenation_flag` is equal to 0 and the picture `currPic` is not the first picture with `nuh_layer_id` equal to `currNuhLayerId` in the bitstream in decoding order, it is a requirement of bitstream conformance that the following constraint applies:

- If the picture `prevNonDiscardablePic` is not associated with a buffering period SEI message that applies the current bitstream partition, the `au_cpb_removal_delay_minus1` of the picture `currPic` shall be equal to the `au_cpb_removal_delay_minus1` of `prevNonDiscardablePic`, indicated in the picture timing SEI message applicable to the current bitstream partition, plus `au_cpb_removal_delay_delta_minus1 + 1`.
- Otherwise, `au_cpb_removal_delay_minus1`, indicated for the picture `currPic` in the picture timing SEI message applicable to the current bitstream partition, shall be equal to `au_cpb_removal_delay_delta_minus1`.

F.14.3.2.3 Picture timing SEI message semantics for multi-layer extensions

The specifications of clause D.3.3 apply with the following modifications and additions:

When the picture timing SEI message is directly contained in a scalable nesting SEI message within an SEI NAL unit with a `nuh_layer_id` value `targetNuhLayerId` greater than 0, the semantics of clause D.3.3 apply to the output bitstream of the independent non-base layer rewriting process of clause F.10.2 with the input variables `assignedBaseLayerId` equal to `nestingLayerIdList[i][0]` and `tIdTarget` equal to `MaxTemporalId[i]` for each value of `i` in the range of 0 to `nestingNumOps – 1`, inclusive.

When the picture timing SEI message is contained in a bitstream partition nesting SEI message, the following applies:

- `frame_field_info_present_flag` is inferred to be equal to 0 for the syntax and semantics of the picture timing SEI message.

NOTE 1 – The frame-field information SEI message can be used to indicate the frame-field information.

- The variables `olsIdx`, `psIdx` and `bspIdx` are set equal to `sei_ols_idx`, `sei_partitioning_scheme_idx` and `bsp_idx`, respectively, of the bitstream partition nesting SEI message containing this buffering period SEI message, and the `bspIdx`-th bitstream partition of the `psIdx`-th partitioning scheme of the `olsIdx`-th OLS is referred to as the current bitstream partition.
- The variable `maxTid` is set equal to `MaxTemporalId[0]` for the scalable nesting SEI message containing the bitstream partition nesting SEI message containing this buffering period SEI message.
- The variable `hrdParamIdx` is set equal to the value of `bsp_hrd_idx[olsIdx][psIdx][maxTid][i][bspIdx]` for any value of `i` in the range of 0 to `BspSchedCnt[olsIdx][psIdx][maxTid] – 1`, inclusive.
- The following applies for the syntax and semantics of the picture timing SEI message:
 - The syntax elements `sub_pic_hrd_params_present_flag`, `sub_pic_cpb_params_in_pic_timing_sei_flag`, `au_cpb_removal_delay_length_minus1`, `dpb_output_delay_length_minus1`, `dpb_output_delay_du_length_minus1`, `du_cpb_removal_delay_increment_length_minus1` and the variable `CpbDpbDelaysPresentFlag` are found in or derived from syntax elements found in the `hrdParamIdx`-th `hrd_parameters()` syntax structure.
 - It is a requirement of bitstream conformance that the values of `sub_pic_hrd_params_present_flag`, `sub_pic_cpb_params_in_pic_timing_sei_flag`, `au_cpb_removal_delay_length_minus1`, `dpb_output_delay_length_minus1`, `dpb_output_delay_du_length_minus1`, `du_cpb_removal_delay_increment_length_minus1` and `CpbDpbDelaysPresentFlag` derived using a particular value of `i` for the derivation of `hrdParamIdx` shall be the same as the value of the respective syntax elements or variable derived using any other allowed value of `i`.

The presence of picture timing SEI messages for the current bitstream partition is specified as follows on the basis of the variable `CpbDpbDelaysPresentFlag` that is derived as specified above:

- If `CpbDpbDelaysPresentFlag` is equal to 1, a picture timing SEI message applicable to the current bitstream partition shall be associated with every access unit in the CVS.
- Otherwise, in the CVS there shall be no access unit that is associated with a picture timing SEI message applicable to the current bitstream partition.

The semantics of `num_decoding_units_minus1` and `num_nalus_in_du_minus1[i]` are replaced with the following:

The variables `targetUnit` and `totalSizeInCtbsY` are derived as follows:

- If the picture timing SEI message is associated with a partition unit, `targetUnit` is set to be the partition unit and the value of `totalSizeInCtbsY` is set equal to the sum of the `PicSizeInCtbsY` of all pictures in the partition unit.
- Otherwise (the picture timing SEI message is associated with an access unit), `targetUnit` is set to be the access unit and the value of `totalSizeInCtbsY` is set equal to the sum of the `PicSizeInCtbsY` of all pictures in the access unit.

`num_decoding_units_minus1` plus 1 specifies the number of decoding units in `targetUnit`. The value of `num_decoding_units_minus1` shall be in the range of 0 to `totalSizeInCtbsY – 1`, inclusive.

`num_nalus_in_du_minus1[i]` plus 1 specifies the number of NAL units in the `i`-th decoding unit of `targetUnit`. The value of `num_nalus_in_du_minus1[i]` shall be in the range of 0 to `totalSizeInCtbsY – 1`, inclusive.

The first decoding unit of `targetUnit` consists of the first `num_nalus_in_du_minus1[0] + 1` consecutive NAL units in decoding order in `targetUnit`. The `i`-th (with `i` greater than 0) decoding unit of `targetUnit` consists of the `num_nalus_in_du_minus1[i] + 1` consecutive NAL units immediately following the last NAL unit in the previous decoding unit of `targetUnit`, in decoding order. There shall be at least one VCL NAL unit in each decoding unit. All non-VCL NAL units associated with a VCL NAL unit shall be included in the same decoding unit as the VCL NAL unit.

F.14.3.2.4 Recovery point SEI message semantics for multi-layer extensions

The recovery point SEI message assists a decoder in determining when the decoding process will produce acceptable pictures in the current layer for display after the decoder initiates random access, layer up-switching or after the encoder indicates a broken link.

When all decoded pictures in earlier access units in decoding order are removed from the bitstream, the recovery point picture (defined below) and all the subsequent pictures in output order in the current layer can be correctly or approximately correctly decoded, the current picture is referred to as a layer random-accessing picture. When all the pictures that belong to the reference layers of the current layer and may be used for reference by the current picture or subsequent pictures in decoding order are correctly decoded, and the recovery point picture and all the subsequent pictures in output order in the current layer can be correctly or approximately correctly decoded when no picture prior to the current picture in decoding order in the current layer are present in the bitstream, the current picture is referred to as a layer up-switching picture.

When the recovery point SEI message applies to the current layer and all the reference layers of the current layer, the current picture is indicated as a layer random-accessing picture. When the recovery point SEI message applies to the current layer but not to all the reference layers of the current layer, the current picture is indicated as a layer up-switching picture.

Decoded pictures in the current layer produced by random access or layer up-switching at or before the current access unit does not need to be correct in content until the indicated recovery point, and the operation of the decoding process for pictures in the current layer starting at the current picture may contain references to pictures unavailable in the decoded picture buffer.

In addition, by use of the `broken_link_flag`, the recovery point SEI message can indicate to the decoder the location of some pictures in the current layer in the bitstream that can result in serious visual artefacts when displayed, even when the decoding process was begun at the location of a previous IRAP access unit in decoding order that contain IRAP pictures in all layers.

NOTE 1 – The `broken_link_flag` can be used by encoders to indicate the location of a point after which the decoding process for the decoding of some pictures in the current layer may cause references to pictures that, though available for use in the decoding process, are not the pictures that were used for reference when the bitstream was originally encoded (e.g., due to a splicing operation performed during the generation of the bitstream).

When the current picture is a layer random access-accessing picture and random access is performed to start decoding from the current access unit, the decoder operates as if the current access unit was the first access unit in the bitstream in decoding order, and the following applies:

- If `poc_msb_cycle_val` is present for the current picture, the `PicOrderCntVal` of the current picture is derived as specified in the process for derivation of `PicOrderCntVal` in clause F.8.3.1.
- Otherwise (`poc_msb_cycle_val` is not present for the current picture), the variable `PrevPicOrderCnt[nuh_layer_id]` used in derivation of `PicOrderCntVal` for each picture in the access unit is set equal to 0.

NOTE 2 – When HRD information is present in the bitstream, a buffering period SEI message should be associated with the access unit associated with the recovery point SEI message in order to establish initialization of the HRD buffer model after a random access.

When the current picture is either a layer random-accessing picture or a layer up-switching picture and layer up-switching is performed to start decoding of the current layer from the current access (while decoding of the reference layers of the current layer have started earlier and pictures of those layers in the current access unit are correctly decoded), the decoder operates as if the current picture was the first picture of the current layer in the bitstream in decoding order, and the `PicOrderCntVal` of the current picture is set equal to the `PicOrderCntVal` of any other decoded picture in the current access unit.

For a particular access unit `auA`, let `auB` be the first access unit that succeeds `auA` in decoding order such that the picture order count values of the pictures in `auB` can be derived without using the picture order count values of pictures preceding `auB` in decoding order. For a particular layer `layerA`, `auA` is not allowed to contain a recovery point SEI message that applies to a set of layers containing at least `layerA` and all reference layers of `layerA` when all of the following conditions are true:

- All pictures in `auA` that are in the reference layers, if any, of `layerA` have both `poc_msb_cycle_val_present_flag` and `poc_reset_idc` equal to 0.
- There is at least one picture `picA` with `poc_msb_cycle_val_present_flag` equal to 1 in the following access units:
 - Access units that follow `auA` in decoding order and precede `auB`, when present, in decoding order.
 - Access unit `auB`, when present and when picture in `auB` with `nuh_layer_id` equal to 0 is not an IRAP picture with `NoCllasOutputFlag` equal to 1.

Any SPS or PPS RBSP that is referred to by a picture of the access unit containing a recovery point SEI message or by any picture in a subsequent access unit in decoding order shall be available to the decoding process prior to its activation, regardless of whether or not the decoding process is started at the beginning of the bitstream or with the access unit, in decoding order, that contains the recovery point SEI message.

recovery_poc_cnt specifies the recovery point of decoded pictures in the current layer in output order. If there is a picture `picB` in the current layer that follows the current picture `picA` but precedes an access unit containing an IRAP picture in the current layer in decoding order and `PicOrderCnt(picB)` is equal to `PicOrderCnt(picA)` plus the value of `recovery_poc_cnt`, where `PicOrderCnt(picA)` and `PicOrderCnt(picB)` are the `PicOrderCntVal` values of `picA` and `picB`, respectively, immediately after the invocation of the decoding process for picture order count for `picB`, the picture `picB` is referred to as the recovery point picture. Otherwise, the first picture `picC` in the current layer in output order for which `PicOrderCnt(picC)` is greater than `PicOrderCnt(picA)` plus the value of `recovery_poc_cnt` is referred to as the recovery point picture, where `PicOrderCnt(picA)` and `PicOrderCnt(picC)` are the `PicOrderCntVal` values of `picA` and `picC`, respectively, immediately after the invocation of the decoding process for picture order count for `picC`. The recovery point picture shall not precede the current picture in decoding order. All decoded pictures in the current layer in output order are

indicated to be correct or approximately correct in content starting at the output order position of the recovery point picture. The value of `recovery_poc_cnt` shall be in the range of $-\text{MaxPicOrderCntLsb} / 2$ to $\text{MaxPicOrderCntLsb} / 2 - 1$, inclusive.

exact_match_flag indicates whether decoded pictures in the current layer at and subsequent to the specified recovery point in output order derived by starting the decoding process at the access unit containing the recovery point SEI message will be an exact match to the pictures in the current layer that would be produced by starting the decoding process at the location of a previous access unit where the picture of the layer in the current layer and the pictures of all the reference layers are IRAP pictures, if any, in the bitstream. The value 0 indicates that the match may not be exact and the value 1 indicates that the match will be exact. When `exact_match_flag` is equal to 1, it is a requirement of bitstream conformance that the decoded pictures in the current layer at and subsequent to the specified recovery point in output order derived by starting the decoding process at the access unit containing the recovery point SEI message shall be an exact match to the pictures in the current layer that would be produced by starting the decoding process at the location of a previous access unit where the picture in the current layer and the pictures of all the reference layers are IRAP pictures, if any, in the bitstream.

NOTE 3 – When performing random access, decoders should infer all references to unavailable pictures as references to pictures containing only intra coding blocks and having sample values given by Y equal to $(1 \ll (\text{BitDepth}_Y - 1))$, Cb and Cr both equal to $(1 \ll (\text{BitDepth}_C - 1))$ (mid-level grey), regardless of the value of `exact_match_flag`.

When `exact_match_flag` is equal to 0, the quality of the approximation at the recovery point is chosen by the encoding process and is not specified in this Specification.

broken_link_flag indicates the presence or absence of a broken link in the current layer at the location of the recovery point SEI message and is assigned further semantics as follows:

- If `broken_link_flag` is equal to 1, pictures in the current layer produced by starting the decoding process at the location of a previous access unit where the picture in the current layer and the pictures of all the reference layers are IRAP pictures may contain undesirable visual artefacts to the extent that decoded pictures in the current layer at and subsequent to the access unit containing the recovery point SEI message in decoding order should not be displayed until the specified recovery point in output order.
- Otherwise (`broken_link_flag` is equal to 0), no indication is given regarding any potential presence of visual artefacts.

When the current picture is a BLA picture, the value of `broken_link_flag` shall be equal to 1.

Regardless of the value of the `broken_link_flag`, pictures in the current layer subsequent to the specified recovery point in output order are specified to be correct or approximately correct in content.

F.14.3.2.5 Structure of pictures information SEI message semantics for multi-layer extensions

The specifications of clause D.3.19 apply by replacing the first paragraph specifying the persistence of the SEI message with the following:

The structure of pictures information SEI message provides information for a list of entries, some of which correspond to the target picture set consists of a series of pictures starting from the current picture until the last picture in decoding order in the current layer in the CLVS or the last picture in decoding order in the current POC resetting period, whichever is earlier.

F.14.3.2.6 Decoding unit information SEI message semantics for multi-layer extensions

The specifications of clause D.3.22 apply with the following modifications and additions:

When the decoding unit information SEI message is directly contained in a scalable nesting SEI message within an SEI NAL unit with a `nuh_layer_id` value `targetNuhLayerId` greater than 0, the semantics of clause D.3.22 apply to the output bitstream of the independent non-base layer rewriting process of clause F.10.2 with the input variables `assignedBaseLayerId` equal to `nestingLayerIdList[i][0]` and `tIdTarget` equal to `MaxTemporalId[i]` for each value of `i` in the range of 0 to `nestingNumOps - 1`, inclusive.

When the decoding unit information SEI message is contained in a bitstream partition nesting SEI message, the following applies:

- The variables `olsIdx`, `psIdx` and `bspIdx` are set equal to `sei_ols_idx`, `sei_partitioning_scheme_idx` and `bsp_idx`, respectively, of the bitstream partition nesting SEI message containing this buffering period SEI message, and the `bspIdx`-th bitstream partition of the `psIdx`-th partitioning scheme of the `olsIdx`-th OLS is referred to as the current bitstream partition.
- The variable `maxTId` is set equal to `MaxTemporalId[0]` for the scalable nesting SEI message containing the bitstream partition nesting SEI message containing this buffering period SEI message.
- The variable `hrdParamIdx` is set equal to the value of `bsp_hrd_idx[olsIdx][psIdx][maxTId][i][bspIdx]` for any value of `i` in the range of 0 to `BspSchedCnt[olsIdx][psIdx][maxTId] - 1`, inclusive.

- The following applies for the syntax and semantics of the decoding unit information SEI message:
 - The syntax elements `sub_pic_hrd_params_present_flag`, `sub_pic_cpb_params_in_pic_timing_sei_flag` and `dpb_output_delay_du_length_minus1`, and the variable `CpbDpbDelaysPresentFlag` are found in or derived from syntax elements in the `hrdParamIdx`-th `hrd_parameters()` syntax structure.
 - It is a requirement of bitstream conformance that the values of `sub_pic_hrd_params_present_flag`, `sub_pic_cpb_params_in_pic_timing_sei_flag` and `dpb_output_delay_du_length_minus1`, and `CpbDpbDelaysPresentFlag` derived using a particular value of `i` for the derivation of `hrdParamIdx` shall be the same as the value of the respective syntax elements or variable derived using any other allowed value of `i`.

The presence of decoding unit information SEI messages for the current bitstream partition is specified as follows on the basis of the syntax elements `sub_pic_hrd_params_present_flag` and `sub_pic_cpb_params_in_pic_timing_sei_flag` and the variable `CpbDpbDelaysPresentFlag` that are found or derived as specified above:

- If `CpbDpbDelaysPresentFlag` is equal to 1, `sub_pic_hrd_params_present_flag` is equal to 1 and `sub_pic_cpb_params_in_pic_timing_sei_flag` is equal to 0, one or more decoding unit information SEI messages applicable to the current bitstream partition shall be associated with each decoding unit of the current bitstream partition in the CVS.
- Otherwise, if `CpbDpbDelaysPresentFlag` is equal to 1, `sub_pic_hrd_params_present_flag` is equal to 1 and `sub_pic_cpb_params_in_pic_timing_sei_flag` is equal to 1, one or more decoding unit information SEI messages applicable to the current bitstream partition may or may not be associated with each decoding unit of the current bitstream partition in the CVS.
- Otherwise (`CpbDpbDelaysPresentFlag` is equal to 0 or `sub_pic_hrd_params_present_flag` is equal to 0), in the CVS there shall be no decoding unit of the current bitstream partition that is associated with a decoding unit information SEI message applicable to the current bitstream partition.

The set of NAL units associated with a decoding unit information SEI message contained in a bitstream partition nesting SEI message consists, in decoding order, of the SEI NAL unit containing the decoding unit information SEI message and all subsequent NAL units in the partition unit up to but not including any subsequent SEI NAL unit containing a decoding unit information SEI message that is contained in a bitstream partition nesting SEI message applying to the current bitstream partition and that has a different value of `decoding_unit_idx`.

It is a requirement of bitstream conformance that all decoding unit information SEI messages that are contained in a bitstream partition nesting SEI message, are associated with the same access unit, apply to the current bitstream partition and have `dpb_output_du_delay_present_flag` equal to 1 shall have the same value of `pic_spt_dpb_output_du_delay`.

F.14.3.2.7 Scalable nesting SEI message semantics for multi-layer extensions

The specifications of clause D.3.24 apply with the following modifications:

It is a requirement of bitstream conformance that the following restrictions apply on containing of SEI messages in a scalable nesting SEI message:

- An SEI message that has `payloadType` equal to 129 (active parameter sets), 132 (decoded picture hash), 133 (scalable nesting), 160 (layers not present), 163 (bitstream partition initial arrival time), 164 (sub-bitstream property) or 166 (overlay information) shall not be directly contained in a scalable nesting SEI message.
- When a scalable nesting SEI message contains a buffering period SEI message, a picture timing SEI message or a decoding unit information SEI message, the scalable nesting SEI message shall not contain any other SEI message with `payloadType` not equal to 0 (buffering period), 1 (picture timing) or 130 (decoding unit information).
- When present, an SEI message that has `payloadType` equal to 162 (bitstream partition nesting) shall be contained within a scalable nesting SEI message.
- When a scalable nesting SEI message contains a bitstream partition nesting SEI message, the scalable nesting SEI message shall not contain any other SEI message.

It is a requirement of bitstream conformance that the following restrictions apply on the value of `bitstream_subset_flag`:

- When the scalable nesting SEI message contains a buffering period SEI message, a picture timing SEI message, a decoding unit information SEI message or a bitstream partition nesting SEI message, `bitstream_subset_flag` shall be equal to 1.
- When the scalable nesting SEI message contains an SEI message that has `payloadType` equal to any value among `SingleLayerSeiList`, 161, 165, 167 or 168, `bitstream_subset_flag` shall be equal to 0.

When `nesting_op_idx[i]` is present, the list `nestingLayerIdList[i]` is set equal to `LayerSetLayerIdList[nesting_op_idx[i]]` derived from the active VPS.

When a scalable nesting SEI message contains a bitstream partition nesting SEI message, the following applies:

- nesting_op_flag shall be equal to 1, default_op_flag shall be equal to 0, nesting_num_ops_minus1 shall be equal to 0 and nesting_op_idx[0] shall not be equal to 0.
- The nuh_layer_id of the SEI NAL unit containing the scalable nesting SEI message shall be equal to the highest value within the list nestingLayerIdList[0].

When a scalable nesting SEI message directly contains a buffering period SEI message, a picture timing SEI message or a decoding unit information SEI message and the nuh_layer_id value of the SEI NAL unit containing the scalable nesting SEI message is greater than 0, it is a requirement of bitstream conformance that for any value of *i* in the range of default_op_flag to nesting_num_ops_minus1, inclusive, the nesting_op_idx[*i*]-th OLS consists of only the layer with nuh_layer_id equal to nestingLayerIdList[*i*][0], the profile of that layer is a profile specified in Annex A and the value of general_inbld_flag (when MaxTemporalId[*i*] is equal to vps_max_sub_layers_minus1) or the value of sub_layer_inbld_flag[tidTarget] (when MaxTemporalId[*i*] is less than vps_max_sub_layers_minus1) in the profile_tier_level() syntax structure associated with that layer is equal to 1.

NOTE – In other words, buffering period, picture timing, or decoding unit information SEI messages that are directly contained in an SEI NAL unit with nuh_layer_id greater than 0 can only be present for rewritable independent non-base layers.

F.14.3.2.8 Region refresh information SEI message semantics for multi-layer extensions

The region refresh information SEI message indicates whether the slice segments that the current SEI message applies to belong to a refreshed region of the current picture.

The variable targetLayerIdList is derived as follows:

- If the region refresh information SEI message applies to the current layer and all the reference layers, targetLayerIdList contains the nuh_layer_id of the current layer and all the reference layers.
- Otherwise, targetLayerIdList contains the nuh_layer_id of the current layer.

The region refresh SEI message is associated with a recovery point SEI message that applies to targetLayerIdList.

A picture that is not an IRAP picture, that belongs to a layer, and that is contained in an access unit containing a recovery point SEI message where the recovery point SEI message applies to that layer is referred to as a gradual decoding refresh (GDR) picture, and the access unit containing the picture is referred to as a GDR access unit. The access unit corresponding to the indicated recovery point picture is referred to as the recovery point access unit.

If there is a picture picB in the current layer that follows the GDR picture picA in the current layer in decoding order in the CVS and PicOrderCnt(picB) is equal to PicOrderCnt(picA) plus the value of recovery_poc_cnt in the recovery point SEI message, where PicOrderCnt(picA) and PicOrderCnt(picB) are the PicOrderCntVal values of picA and picB, respectively, immediately after the invocation of the decoding process for picture order count for picB, let the variable lastPicInSet be the recovery point picture. Otherwise, let lastPicInSet be the picture in targetLayerIdList that immediately precedes the recovery point picture in output order. The picture lastPicInSet shall not precede the GDR access unit in decoding order.

Let gdrPicSet be the set of pictures in targetLayerIdList starting from a GDR access unit to the access unit containing lastPicInSet, inclusive, in output order. When the decoding process for the current layer is started from a GDR access unit, the refreshed region in each picture of the gdrPicSet is indicated to be the region of the picture that is correct or approximately correct in content, and, when lastPicInSet is contained in the recovery point access unit, the refreshed region in lastPicInSet covers the entire picture.

The slice segments of the current picture to which a region refresh information SEI message applies consist of all slice segments within the picture that follow the SEI NAL unit containing the region refresh information SEI message and precede the next SEI NAL unit, in decoding order, containing a region refresh information SEI message (if any) that has the same targetLayerIdList as the current SEI message. These slice segments are referred to as the slice segments associated with the region refresh information SEI message.

Let gdrAuSet be the set of access units corresponding to gdrPicSet. A gdrAuSet and the corresponding gdrPicSet are referred to as being associated with the recovery point SEI message contained in the GDR access unit.

Region refresh information SEI messages shall not be present in an access unit unless the access unit is included in a gdrAuSet associated with a recovery point SEI message. When any picture that is included in a gdrPicSet is associated with one or more region refresh information SEI messages, all pictures in the gdrPicSet shall be associated with one or more region refresh information SEI messages.

refreshed_region_flag equal to 1 indicates that the slice segments associated with the current SEI message belong to the refreshed region in the current picture. refreshed_region_flag equal to 0 indicates that the slice segments associated with the current SEI message may not belong to the refreshed region in the current picture.

When one or more region refresh information SEI messages are associated with a picture belonging to `gdrPicSet` and the first slice segment of the picture in decoding order does not have an associated region refresh information SEI message, the value of `refreshed_region_flag` for the slice segments of the picture that precede the first region refresh information SEI message is inferred to be equal to 0.

When `lastPicInSet` is the recovery point picture, and any region refresh SEI message is associated with the recovery point access unit, the first slice segment of the picture in decoding order shall have an associated region refresh SEI message, and the value of `refreshed_region_flag` shall be equal to 1 in all region refresh SEI messages associated with the picture.

When one or more region refresh information SEI messages are associated with a picture, the refreshed region in the picture is specified as the set of CTUs in all slice segments of the picture that are associated with region refresh information SEI messages that have `refreshed_region_flag` equal to 1. Other slice segments belong to the non-refreshed region of the picture.

It is a requirement of bitstream conformance that when a dependent slice segment belongs to the refreshed region, the preceding slice segment in decoding order shall also belong to the refreshed region.

Let `gdrRefreshedSliceSegmentSet` be the set of all slice segments that belong to the refreshed regions in the `gdrPicSet`. The variable `upSwitchingRefreshedSliceSegmentSet` is derived as follows:

- If `targetLayerIdList` contains only one non-zero `nuh_layer_id`, `upSwitchingRefreshedSliceSegmentSet` is defined as the set inclusive of the following:
 - all slice segments of all pictures of the reference layers that precede, in decoding order, the current picture and that may be used for reference by the current picture or subsequent pictures of the reference layers.
 - all slice segments of all pictures of the reference layers that succeed, in decoding order, the current picture and that belong to `gdrAuSet`.
- Otherwise, `upSwitchingRefreshedSliceSegmentSet` is defined as an empty set.

When a `gdrPicSet` contains one or more pictures associated with region refresh information SEI messages, it is a requirement of bitstream conformance that the following constraints all apply:

- For each layer in `targetLayerIdList`, the refreshed region in the first picture, in decoding order, that belongs to the layer and that is included in `gdrPicSet` that contains any refreshed region shall contain only coding units that are coded in an intra coding mode or inter-layer prediction from slice segments belonging to the union of `gdrRefreshedSliceSegmentSet` and `upSwitchingRefreshedSliceSegmentSet`.
- For each picture included in the `gdrPicSet`, the syntax elements in `gdrRefreshedSliceSegmentSet` shall be constrained such that no samples or motion vector values outside of the union of `gdrRefreshedSliceSegmentSet` and `upSwitchingRefreshedSliceSegmentSet` are used for inter prediction or inter-layer prediction in the decoding process of any samples within `gdrRefreshedSliceSegmentSet`.
- For any picture that follows the picture `lastPicInSet` in output order, the syntax elements in the slice segments of the picture shall be constrained such that no samples or motion vector values outside of the union of `gdrRefreshedSliceSegmentSet` and `upSwitchingRefreshedSliceSegmentSet` are used for inter prediction or inter-layer prediction in the decoding process of the picture other than those of the other pictures that follow the picture `lastPicInSet` in output order.

F.14.3.2.9 Coded region completion SEI message semantics for multi-layer extensions

The specifications of clause D.3.37 apply with the following additions:

The `nuh_layer_id` value of the SEI NAL unit containing a coded region completion SEI message shall be equal to the `nuh_layer_id` value of the associated VCL NAL unit for the SEI NAL unit.

When an access unit contains one or more layers not present SEI messages, any coded region completion SEI message with `next_segment_address` equal to 0 in that access unit shall follow the first layers not present SEI message, in decoding order, that is present in that access unit.

NOTE – The specifications given in clause F.7.4.2.4.4 for associating NAL units to access units facilitate determining of an end of an access unit when the first NAL unit of the next access unit is available. Some implementations may operate on access unit basis, e.g. when inputting NAL units to a decoder. The latency in such implementations is potentially reduced, when a bitstream includes multiple layers and an end of an access unit for the OLS with index `trgtOlsIdx` is concluded as follows in response to decoding a coded region completion SEI message with `next_segment_address` equal to 0:

- The variable `layerMayBePresent[layerIdx]` is derived as follows:
 - If a layers not present SEI message is present in the current CVS and precedes the coded region completion SEI message in decoding order, `layerMayBePresentFlag[layerIdx]` is set equal to `!layer_not_present_flag[layerIdx]` of the previous layers not present SEI message, in decoding order, for each value of `layerIdx` from 0 to `MaxLayersMinus1`, inclusive.
 - Otherwise, `layerMayBePresentFlag[layerIdx]` is set equal to 1 for each value of `layerIdx` from 0 to `MaxLayersMinus1`, inclusive.

- Let a set of `nuh_layer_id` values `trgtLayerIdList` of the necessary layers for an OLS with index `trgtOlsIdx` be equal to `TargetDecLayerIdList` derived using Equation F-57 by setting `TargetOlsIdx` equal to `trgtOlsIdx`.
- Let `numTrgtLayerIds` be equal to the number of `nuh_layer_id` values in `trgtLayerIdList`.
- Let `currLayerId` be equal to the `nuh_layer_id` value of the VCL NAL unit associated with the SEI NAL unit containing this coded region completion SEI message.
- Let `prevNecessaryLayerTrgtIdx` be the index of the greatest value within `trgtLayerIdList` that is less than or equal to `currLayerId`.
- When one of the following is true, there are no VCL NAL units of any necessary layer of the OLS with index `trgtOlsIdx` following, in decoding order, the VCL NAL unit associated with the SEI NAL unit containing this SEI message within the current access unit:
 - `currLayerId` is greater than or equal to any value in `trgtLayerIdList`.
 - `layerMayBePresentFlag[LayerIdxInVps[layerId]]` is equal to 0 for all the values of `layerId` equal to `trgtLayerIdList[trgtIdx]` for each `trgtIdx` in the range of `prevNecessaryLayerTrgtIdx + 1` to `numTrgtLayerIds - 1`, inclusive.

F.14.3.3 Layers not present SEI message semantics

The layers not present SEI message provides a mechanism for signalling that VCL NAL units of particular layers indicated by the VPS are not present in a particular set of access units.

The target access units are defined as the set of access units starting from the access unit containing the layers not present SEI message up to but not including the next access unit, in decoding order, that contains a layers not present SEI message or the end of the CVS, whichever is earlier in decoding order.

When present, the layers not present SEI message applies to the target access units.

When a layers not present SEI message is associated with a coded picture with TemporalId `firstTid` that is greater than 0 and that coded picture is followed, in decoding order, by any coded picture with TemporalId less than `firstTid` in the same CVS, a layers not present SEI message shall be present for the next coded picture, in decoding order, with TemporalId less than `firstTid`.

lnp_sei_active_vps_id identifies the active VPS of the CVS containing the layers not present SEI message. The value of **lnp_sei_active_vps_id** shall be equal to the value of **vps_video_parameter_set_id** of the active VPS for the VCL NAL units of the access unit containing the SEI message.

layer_not_present_flag[i] equal to 1 indicates that there are no VCL NAL units with `nuh_layer_id` equal to `layer_id_in_nuh[i]` present in the target access units. **layer_not_present_flag[i]** equal to 0 indicates that there may or may not be VCL NAL units with `nuh_layer_id` equal to `layer_id_in_nuh[i]` present in the target access units.

When **layer_not_present_flag[i]** is equal to 1 and `i` is less than `MaxLayersMinus1`, **layer_not_present_flag[LayerIdxInVps[IdPredictedLayer[layer_id_in_nuh[i]][j]]]** shall be equal to 1 for all values of `j` in the range of 0 to `NumPredictedLayers[layer_id_in_nuh[i]] - 1`, inclusive.

F.14.3.4 Inter-layer constrained tile sets SEI message semantics

The scope of the inter-layer constrained tile sets SEI message is a complete CLVS of the layer with `nuh_layer_id` equal to `targetLayerId`. When an inter-layer constrained tile sets SEI message is present for any coded picture of a CLVS and the first coded picture of the CLVS in decoding order is an IRAP picture, the inter-layer constrained tile sets SEI message shall be present for the first coded picture of the CLVS in decoding order and may also be present for other coded pictures of the CLVS.

The inter-layer constrained tile sets SEI message shall not be present for the layer with `nuh_layer_id` equal to `targetLayerId` when **tiles_enabled_flag** is equal to 0 for any PPS that is active for the pictures of the CLVS of the layer with `nuh_layer_id` equal to `targetLayerId`.

The inter-layer constrained tile sets SEI message shall not be present for the layer with `nuh_layer_id` equal to `targetLayerId` unless every PPS that is active for the pictures of the CLVS of the layer with `nuh_layer_id` equal to `targetLayerId` has **tile_boundaries_aligned_flag** equal to 1 or fulfills the conditions that would be indicated by **tile_boundaries_aligned_flag** being equal to 1.

An identified tile set is defined as a set of tiles within a picture with `nuh_layer_id` equal to `targetLayerId` that contains the tiles specified below on the basis of the values of the syntax elements contained in the inter-layer constrained tile sets SEI message.

An associated reference tile set for the `iPred`-th identified tile set with **ilcts_id[iPred]** equal to `ilctsId`, if any, is defined as the identified tile set present in the same access unit as the `iPred`-th identified tile set and associated with **ilcts_id[iRef]** equal to `ilctsId` and `nuh_layer_id` equal to `IdDirectRefLayer[targetLayerId][j]` for any value of `j` in the range of 0 to `NumDirectRefLayers[targetLayerId] - 1`, inclusive.

The presence of the inter-layer constrained tile sets SEI message indicates that the inter-layer prediction process is constrained such that no sample value outside each associated reference tile set, and no sample value at a fractional sample position that is derived using one or more sample values outside each associated reference tile set, is used for inter-layer prediction of any sample within the identified tile set.

NOTE 1 – When loop filtering and resampling filter are applied across tile boundaries, inter-layer prediction of any samples within an identified tile set that refers to samples within 8 samples from an associated reference tile set boundary that is not also a picture boundary may result in propagation of mismatch error. An encoder can avoid such potential error propagation by avoiding the use of motion vectors that cause such references.

When more than one inter-layer constrained tile sets SEI message is present for a CLVS, they shall contain identical content.

The number of inter-layer constrained tile sets SEI messages for the same CLVS in each access unit shall not exceed 5.

il_all_tiles_exact_sample_value_match_flag equal to 1 indicates that, within the CLVS, when the CTBs that are outside of any tile in any identified tile set specified in this inter-layer constrained tile sets SEI message are not decoded and the boundaries of the tile is treated as picture boundaries for purposes of the decoding process, the value of each sample in the tile would be exactly the same as the value of the sample that would be obtained when all the CTBs of all pictures in the CLVS are decoded. **il_all_tiles_exact_sample_value_match_flag** equal to 0 indicates that, within the CLVS, when the CTBs that are outside of any tile in any identified tile set specified in this inter-layer constrained tile sets SEI message are not decoded and the boundaries of the tile is treated as picture boundaries for purposes of the decoding process, the value of each sample in the tile may or may not be exactly the same as the value of the same sample when all the CTBs of all pictures in the CLVS are decoded.

il_one_tile_per_tile_set_flag equal to 1 indicates that each identified tile set contains one tile and **il_num_sets_in_message_minus1** is not present.

It is a requirement of bitstream conformance that when **il_one_tile_per_tile_set_flag** is equal to 1, the value of $(\text{num_tile_columns_minus1} + 1) * (\text{num_tile_rows_minus1} + 1)$ shall be less than or equal to 256.

When **il_one_tile_per_tile_set_flag** is equal to 0, the identified tile sets are signalled explicitly.

il_num_sets_in_message_minus1 plus 1 specifies the number of the identified tile sets in the SEI message. The value of **il_num_sets_in_message_minus1** shall be in the range of 0 to 255, inclusive.

skipped_tile_set_present_flag equal to 1 indicates that, within the CLVS, the **il_num_sets_in_message_minus1**-th identified tile set consists of those remaining tiles that are not included in any earlier identified tile sets in the same message and all the prediction blocks that are inside the **il_num_sets_in_message_minus1**-th identified tile set are inter-layer predicted from inter-layer reference pictures with **nuh_layer_id** equal to **IdDirectRefLayer[targetLayerId][NumDirectRefLayers[targetLayerId] - 1]** and no residual coding() syntax structure is present in any transform unit of the **il_num_sets_in_message_minus1**-th identified tile set. **skipped_tile_set_present_flag** equal to 0 does not indicate a bitstream constraint within the CLVS. When not present, the value of **skipped_tile_set_present_flag** is inferred to be equal to 0.

ilcts_id[i] contains an identifying number that may be used to identify the purpose of the i-th identified tile set (for example, to identify an area to be extracted from the coded video sequence for a particular purpose). The value of **ilcts_id[i]** shall be in the range of 0 to $2^{32} - 2$, inclusive.

Values of **ilcts_id[i]** from 0 to 255, inclusive, and from $2^{31} - 1$, inclusive, may be used as determined by the application. If **il_one_tile_per_tile_set_flag** is equal to 1, values of **ilcts_id[i]** from 256 to 511, inclusive, are used for the inferred **ilcts_id[i]** values as specified above. Otherwise, values of **ilcts_id[i]** from 256 to 511, inclusive, are reserved for future use by ITU-T | ISO/IEC. Values of **ilcts_id[i]** from 2^{31} to $2^{32} - 2$, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering an indicated value of **ilcts_id[i]** in the range of 256 to 511, inclusive, or in the range of 2^{31} to $2^{32} - 2$, inclusive, shall ignore it.

When **ilcts_id[i]** is not present for i in the range of 0 to $(\text{num_tile_columns_minus1} + 1) * (\text{num_tile_rows_minus1} + 1) - 1$, inclusive, due to **il_one_tile_per_tile_set_flag** being equal to 1, it is inferred to be equal to $256 + i$.

il_num_tile_rects_in_set_minus1[i] plus 1 specifies the number of rectangular regions of tiles in the i-th identified tile set. The value of **il_num_tile_rects_in_set_minus1[i]** shall be in the range of 0 to $(\text{num_tile_columns_minus1} + 1) * (\text{num_tile_rows_minus1} + 1) - 1$, inclusive.

il_top_left_tile_idx[i][j] and **il_bottom_right_tile_idx[i][j]** identify the tile position of the top-left tile and the tile position of the bottom-right tile in a rectangular region of the i-th identified tile set, respectively, in tile raster scan order.

ilc_idc[i] equal to 1 indicates that no samples outside of any associated reference tile set and no samples at a fractional sample position that is derived using one or more samples outside of any associated reference tile set are used for inter-layer prediction of any sample within the i-th identified tile set. **ilc_idc[i]** equal to 2 indicates that no prediction block in

the *i*-th identified tile set is predicted from an inter-layer reference picture. *ilc_idc*[*i*] equal to 0 indicates that the inter-layer prediction process may or may not be constrained for the prediction block in the *i*-th identified tile set. The value of *ilc_idc*[*i*] equal to 3 is reserved.

il_exact_sample_value_match_flag[*i*] equal to 1 indicates that, within the CLVS, when the CTBs that do not belong to the *i*-th identified tile set are not decoded and the boundaries of the *i*-th identified tile set are treated as picture boundaries for the purposes of the decoding process, the value of each sample in the *i*-th identified tile set would be exactly the same as the value of the sample that would be obtained when all the CTBs of all pictures in the CLVS are decoded. **il_exact_sample_value_match_flag**[*i*] equal to 0 indicates that, within the CLVS, when the CTBs that are outside of the *i*-th identified tile set are not decoded and the boundaries of the *i*-th identified tile set are treated as picture boundaries for the purposes of the decoding process, the value of each sample in the *i*-th identified tile set may or may not be exactly the same as the value of the same sample when all the CTBs of all pictures in the CLVS are decoded.

NOTE 2 – It is feasible to use **il_exact_sample_value_match_flag** equal to 1 when using certain combinations of **loop_filter_across_tiles_enabled_flag**, **pps_loop_filter_across_slices_enabled_flag**, **pps_deblocking_filter_disabled_flag**, **slice_loop_filter_across_slices_enabled_flag**, **slice_deblocking_filter_disabled_flag**, **sample_adaptive_offset_enabled_flag**, **slice_sao_luma_flag** and **slice_sao_chroma_flag**.

all_tiles_ilc_idc equal to 1 indicates that, for each identified tile set within the CLVS, no sample value outside of each associated reference tile set and no sample value at a fractional sample position that is derived using one or more samples outside of each associated reference tile set is used for inter-layer prediction of any sample within the identified tile set. **all_tiles_ilc_idc** equal to 2 indicates that, within the CLVS, no prediction block in each identified tile set is predicted from an inter-layer reference picture. **all_tiles_ilc_idc** equal to 0 indicates that, within the CLVS, the inter-layer prediction process may or may not be constrained for the identified tile sets. The value of **all_tiles_ilc_idc** equal to 3 is reserved. When **all_tiles_ilc_idc** is not present, it is inferred to be equal to 0.

F.14.3.5 Bitstream partition nesting SEI message semantics

The bitstream partition nesting SEI message provides a mechanism to associate SEI messages with a bitstream partition of a partitioning scheme of an OLS.

NOTE – The bitstream partition nesting SEI message must be contained within a scalable nesting SEI message. Constraints on the scalable nesting SEI message containing a bitstream partition nesting SEI message are specified in clause F.14.3.2.7.

A bitstream partition nesting SEI message contains one or more SEI messages.

sei_ols_idx specifies the index of the OLS to which the contained SEI messages apply. The value of **sei_ols_idx** shall be in the range of 0 to **NumOutputLayerSets** – 1, inclusive.

It is a requirement of bitstream conformance that **OlsIdxToLsIdx**[**sei_ols_idx**] shall be equal to **nesting_op_idx**[0] of the scalable nesting SEI message that contains the bitstream partition nesting SEI message.

sei_partitioning_scheme_idx specifies the index of the partitioning scheme to which the contained SEI messages apply. The value of **sei_partitioning_scheme_idx** shall be in the range of 0 to **num_signalled_partitioning_schemes**[**sei_ols_idx**], inclusive.

bsp_idx specifies the index of the bitstream partition to which the contained SEI messages apply. The value of **bsp_idx** shall be in the range of 0 to **num_partitions_in_scheme_minus1**[**sei_ols_idx**][**sei_partitioning_scheme_idx**], inclusive.

num_seis_in_bsp_minus1 plus 1 specifies the number of **sei_message**() syntax structures contained in the **bsp_nesting**() syntax structure. The value of **num_seis_in_bsp_minus1** shall be in the range of 0 to 63, inclusive.

bsp_nesting_zero_bit shall be equal to 0.

F.14.3.6 Bitstream partition initial arrival time SEI message semantics

The bitstream partition initial arrival time SEI message specifies the initial arrival times to be used in the bitstream-partition-specific CPB operation.

When present, this SEI message shall be contained within a bitstream partition nesting SEI message that is contained in a scalable nesting SEI message, and the same bitstream partition nesting SEI message shall also contain a buffering period SEI message.

The following applies for each value of *i* in the range of 0 to **BspSchedCnt**[**olsIdx**][**psIdx**][**MaxTemporalId**[0]] – 1, inclusive:

- The variable **hrdParamIdx**[*i*] is set equal to the value of **bsp_hrd_idx**[**olsIdx**][**psIdx**][**maxTemporalId**[0]][*i*][**bspIdx**], where **olsIdx**, **psIdx** and **bspIdx** are equal to **sei_ols_idx**, **sei_partitioning_scheme_idx** and **bsp_idx**, respectively, of the bitstream partition nesting SEI message containing this bitstream partition initial arrival time SEI message, and **maxTemporalId**[0] is the value of

MaxTemporalId[0] for the scalable nesting SEI message containing the bitstream partition nesting SEI message containing this bitstream partition initial arrival time SEI message.

- The variable initialCpbRemovalDelayLength[i] is set equal to initial_cpb_removal_delay_length_minus1 + 1, where initial_cpb_removal_delay_length_minus1 is found in the hrdParamIdx[i]-th hrd_parameters() syntax structure in the active VPS.
- The variable nalHrdParamsPresent[i] is set equal to the value of nal_hrd_parameters_present_flag in the hrdParamIdx[i]-th hrd_parameters() syntax structure in the active VPS.
- The variable vclHrdParamsPresent[i] is set equal to the value of vcl_hrd_parameters_present_flag in the hrdParamIdx[i]-th hrd_parameters() syntax structure in the active VPS.

It is a requirement of bitstream conformance that the value of nalHrdParamsPresent[i] shall be the same for all values of i in the range of 0 to BspSchedCnt[olsIdx][psIdx][MaxTemporalId[0]] – 1, inclusive.

It is a requirement of bitstream conformance that the value of vclHrdParamsPresent[i] shall be the same for all values of i in the range of 0 to BspSchedCnt[olsIdx][psIdx][MaxTemporalId[0]] – 1, inclusive.

The variable nalInitialArrivalDelayPresent is set equal to nalHrdParamsPresent[i] of any value of i in the range of 0 to BspSchedCnt[olsIdx][psIdx][MaxTemporalId[0]] – 1, inclusive.

The variable vclInitialArrivalDelayPresent is set equal to vclHrdParamsPresent[i] of any value of i in the range of 0 to BspSchedCnt[olsIdx][psIdx][MaxTemporalId[0]] – 1, inclusive.

nal_initial_arrival_delay[i] specifies the initial arrival time for the i-th delivery schedule of the bitstream partition to which this SEI message applies, when NAL HRD parameters are in use. The length, in bits, of the nal_initial_arrival_delay[i] syntax element is equal to initialCpbRemovalDelayLength[i].

vcl_initial_arrival_delay[i] specifies the initial arrival time for the i-th delivery schedule of the bitstream partition to which this SEI message applies, when VCL HRD parameters are in use. The length, in bits, of the vcl_initial_arrival_delay[i] syntax element is equal to initialCpbRemovalDelayLength[i].

F.14.3.7 Sub-bitstream property SEI message semantics

The sub-bitstream property SEI message, when present, provides the bit rate information for a sub-bitstream created by discarding those pictures in the layers that do not belong to the output layers of the OLSs specified by the active VPS and that do not affect the decoding of the output layers.

When present, the sub-bitstream property SEI message shall be associated with an initial IRAP access unit and the information provided by the SEI messages applies to the bitstream corresponding to the CVS containing the associated initial IRAP access unit.

sb_property_active_vps_id identifies the active VPS. The value of sb_property_active_vps_id shall be equal to the value of vps_video_parameter_set_id of the active VPS referred to by the VCL NAL units of the associated access unit.

num_additional_sub_streams_minus1 plus 1 specifies the number of the sub-bitstreams for which the bit rate information may be provided by this SEI message. The value of num_additional_sub_streams_minus1 shall be in the range of 0 to $2^{10} - 1$, inclusive.

sub_bitstream_mode[i] specifies how the i-th sub-bitstream is generated. The value of sub_bitstream_mode[i] shall be equal to 0 or 1, inclusive. The values 2 and 3 are reserved for future use by ITU-T and ISO/IEC. When sub_bitstream_mode[i] is the greater than 1, decoders shall ignore the syntax elements ols_idx_to_vps[i], highest_sublayer_id[i], avg_sb_property_bit_rate[i] and max_sb_property_bit_rate[i].

When sub_bitstream_mode[i] is equal to 0, the i-th sub-bitstream is generated as follows:

- Let lsIdx be equal to OlsIdxToLsIdx[ols_idx_to_vps[i]].
- A sub-bitstream subBitstream[i] is first created as follows:
 - If lsIdx is less than or equal to vps_num_layer_sets_minus1 and vps_base_layer_internal_flag is equal to 1, the sub-bitstream extraction process as specified in clause 10 is invoked with the bitstream corresponding to the CVS containing the sub-bitstream property SEI message, highest_sublayer_id[i] and LayerSetLayerIdList[lsIdx] as inputs, and the output is assigned to subBitstream[i].
 - Otherwise, if lsIdx is less than or equal to vps_num_layer_sets_minus1 and vps_base_layer_internal_flag is equal to 0, the sub-bitstream extraction process as specified in clause F.10.1 is invoked with the bitstream corresponding to the CVS containing the sub-bitstream property SEI message, highest_sublayer_id[i] and LayerSetLayerIdList[lsIdx] as inputs, and the output is assigned to subBitstream[i].

- Otherwise, the sub-bitstream extraction process as specified in clause F.10.3 is invoked with the bitstream corresponding to the CVS containing the sub-bitstream property SEI message, `highest_sublayer_id[i]` and `LayerSetLayerIdList[lIdx]` as inputs, and the output is assigned to `subBitstream[i]`.
- Remove all NAL units for which the `nuh_layer_id` is not included in `TargetOptLayerIdList` and either of the following conditions is true:
 - The value of `nal_unit_type` is not in the range of `BLA_W_LP` to `RSV_IRAP_VCL23`, inclusive, and `max_tid_il_ref_pics_plus1[LayerIdxInVps[nuh_layer_id]][LayerIdxInVps[layerId]]` is equal to 0 for `layerId` values included in `TargetOptLayerIdList`.
 - `TemporalId` is greater than the maximum value of `max_tid_il_ref_pics_plus1[LayerIdxInVps[nuh_layer_id]][LayerIdxInVps[layerId]] - 1` for all `layerId` values included in `TargetOptLayerIdList`.

When `sub_bitstream_mode[i]` is equal to 1, the *i*-th sub-bitstream is generated as specified by the above process followed by:

- Remove all NAL units with `nuh_layer_id` not among the values included in `TargetOptLayerIdList` and with `discardable_flag` equal to 1.

`ols_idx_to_vps[i]` specifies the index of the OLS corresponding to the *i*-th sub-bitstream. The value of `ols_idx_to_vps[i]` shall be in the range of 0 to `NumOutputLayerSets - 1`, inclusive.

`highest_sublayer_id[i]` specifies the highest `TemporalId` of access units in the *i*-th sub-bitstream.

`avg_sb_property_bit_rate[i]` indicates the average bit rate of the *i*-th sub-bitstream, in bits per second. The value is given by `BitRateBPS(avg_sb_property_bit_rate[i])` with the function `BitRateBPS()` being specified as follows:

$$\text{BitRateBPS}(x) = (x \& (2^{14} - 1)) * 10^{(2 + (x \gg 14))} \quad (\text{F-91})$$

The average bit rate is derived according to the access unit removal time specified in clause F.13. In the following, `bTotal` is the number of bits in all NAL units of the *i*-th sub-bitstream, `t1` is the removal time (in seconds) of the first access unit to which the VPS applies and `t2` is the removal time (in seconds) of the last access unit (in decoding order) to which the VPS applies. With `x` specifying the value of `avg_sb_property_bit_rate[i]`, the following applies:

- If `t1` is not equal to `t2`, the following condition shall be true:

$$(x \& (2^{14} - 1)) == \text{Round}(bTotal \div ((t_2 - t_1) * 10^{(2 + (x \gg 14))})) \quad (\text{F-92})$$

- Otherwise (`t1` is equal to `t2`), the following condition shall be true:

$$(x \& (2^{14} - 1)) == 0 \quad (\text{F-93})$$

`max_sb_property_bit_rate[i]` indicates an upper bound for the bit rate of the *i*-th sub-bitstream in any one-second time window of access unit removal time as specified in clause F.13. The upper bound for the bit rate in bits per second is given by `BitRateBPS(max_sb_property_bit_rate[i])`. The bit rate values are derived according to the access unit removal time specified in clause F.13. In the following, `t1` is any point in time (in seconds), `t2` is set equal to `t1 + 1 ÷ 100` and `bTotal` is the number of bits in all NAL units of access units with a removal time greater than or equal to `t1` and less than `t2`. With `x` specifying the value of `max_sb_property_bit_rate[i]`, the following condition shall be obeyed for all values of `t1`:

$$(x \& (2^{14} - 1)) \geq bTotal \div ((t_2 - t_1) * 10^{(2 + (x \gg 14))}) \quad (\text{F-94})$$

F.14.3.8 Alpha channel information SEI message semantics

The alpha channel information SEI message provides information about alpha channel sample values and post-processing applied to the decoded alpha planes coded in auxiliary pictures of type `AUX_ALPHA` and one or more associated primary pictures.

For an auxiliary picture with `nuh_layer_id` equal to `nuhLayerIdA` and `AuxId[nuhLayerIdA]` equal to `AUX_ALPHA`, an associated primary picture, if any, is a picture in the same access unit having `AuxId[nuhLayerIdB]` equal to 0 such that `ScalabilityId[LayerIdxInVps[nuhLayerIdA]][j]` is equal to `ScalabilityId[LayerIdxInVps[nuhLayerIdB]][j]` for all values of `j` in the range of 0 to 2, inclusive, and 4 to 15, inclusive.

When an access unit contains an auxiliary picture `picA` with `nuh_layer_id` equal to `nuhLayerIdA` and `AuxId[nuhLayerIdA]` equal to `AUX_ALPHA`, the alpha channel sample values of `picA` persist in output order until one or more of the following conditions are true:

- The next picture, in output order, with `nuh_layer_id` equal to `nuhLayerIdA` is output.
- A CLVS containing the auxiliary picture `picA` ends.
- The bitstream ends.

- A CLVS of any associated primary layer of the auxiliary picture layer with `nuh_layer_id` equal to `nuhLayerIdA` ends.

The following semantics apply separately to each `nuh_layer_id` `targetLayerId` among the `nuh_layer_id` values to which the alpha channel information SEI message applies.

alpha_channel_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous alpha channel information SEI message in output order that applies to the current layer. `alpha_channel_cancel_flag` equal to 0 indicates that alpha channel information follows.

Let `currPic` be the picture that the alpha channel information SEI message is associated with. The semantics of alpha channel information SEI message persist for the current layer in output order until one or more of the following conditions are true:

- A new CLVS of the current layer begins.
- The bitstream ends.
- A picture `picB` with `nuh_layer_id` equal to `targetLayerId` in an access unit containing an alpha channel information SEI message with `nuh_layer_id` equal to `targetLayerId` is output having `PicOrderCnt(picB)` greater than `PicOrderCnt(currPic)`, where `PicOrderCnt(picB)` and `PicOrderCnt(currPic)` are the `PicOrderCntVal` values of `picB` and `currPic`, respectively, immediately after the invocation of the decoding process for picture order count for `picB`.

alpha_channel_use_idc equal to 0 indicates that for alpha blending purposes the decoded samples of the associated primary picture should be multiplied by the interpretation sample values of the auxiliary coded picture in the display process after output from the decoding process. `alpha_channel_use_idc` equal to 1 indicates that for alpha blending purposes the decoded samples of the associated primary picture should not be multiplied by the interpretation sample values of the auxiliary coded picture in the display process after output from the decoding process. `alpha_channel_use_idc` equal to 2 indicates that the usage of the auxiliary picture is unspecified. Values greater than 2 for `alpha_channel_use_idc` are reserved for future use by ITU-T | ISO/IEC. When not present, the value of `alpha_channel_use_idc` is inferred to be equal to 2.

alpha_channel_bit_depth_minus8 plus 8 specifies the bit depth of the samples of the luma sample array of the auxiliary picture. `alpha_channel_bit_depth_minus8` shall be in the range 0 to 7 inclusive. `alpha_channel_bit_depth_minus8` shall be equal to `bit_depth_luma_minus8` of the associated primary picture.

alpha_transparent_value specifies the interpretation sample value of an auxiliary coded picture luma sample for which the associated luma and chroma samples of the primary coded picture are considered transparent for purposes of alpha blending. The number of bits used for the representation of the `alpha_transparent_value` syntax element is `alpha_channel_bit_depth_minus8 + 9`.

alpha_opaque_value specifies the interpretation sample value of an auxiliary coded picture luma sample for which the associated luma and chroma samples of the primary coded picture are considered opaque for purposes of alpha blending. The number of bits used for the representation of the `alpha_opaque_value` syntax element is `alpha_channel_bit_depth_minus8 + 9`.

alpha_channel_incr_flag equal to 0 indicates that the interpretation sample value for each decoded auxiliary picture luma sample value is equal to the decoded auxiliary picture sample value for purposes of alpha blending. `alpha_channel_incr_flag` equal to 1 indicates that, for purposes of alpha blending, after decoding the auxiliary picture samples, any auxiliary picture luma sample value that is greater than $\text{Min}(\text{alpha_opaque_value}, \text{alpha_transparent_value})$ should be increased by one to obtain the interpretation sample value for the auxiliary picture sample and any auxiliary picture luma sample value that is less than or equal to $\text{Min}(\text{alpha_opaque_value}, \text{alpha_transparent_value})$ should be used, without alteration, as the interpretation sample value for the decoded auxiliary picture sample value. When not present, the value of `alpha_channel_incr_flag` is inferred to be equal to 0.

alpha_channel_clip_flag equal to 0 indicates that no clipping operation is applied to obtain the interpretation sample values of the decoded auxiliary picture. `alpha_channel_clip_flag` equal to 1 indicates that the interpretation sample values of the decoded auxiliary picture are altered according to the clipping process described by the `alpha_channel_clip_type_flag` syntax element. When not present, the value of `alpha_channel_clip_flag` is inferred to be equal to 0.

alpha_channel_clip_type_flag equal to 0 indicates that, for purposes of alpha blending, after decoding the auxiliary picture samples, any auxiliary picture luma sample that is greater than $(\text{alpha_opaque_value} - \text{alpha_transparent_value}) / 2$ is set equal to `alpha_opaque_value` to obtain the interpretation sample value for the auxiliary picture luma sample and any auxiliary picture luma sample that is less or equal than $(\text{alpha_opaque_value} - \text{alpha_transparent_value}) / 2$ is set equal to `alpha_transparent_value` to obtain the interpretation sample value for the auxiliary picture luma sample. `alpha_channel_clip_type_flag` equal to 1 indicates that, for purposes of alpha blending, after decoding the auxiliary picture samples, any auxiliary picture luma sample that is greater than `alpha_opaque_value` is set equal to `alpha_opaque_value` to obtain the interpretation sample value for the auxiliary picture luma sample and any auxiliary picture luma sample that is less than or equal to `alpha_transparent_value` is set equal to

alpha_transparent_value to obtain the interpretation sample value for the auxiliary picture luma sample.

NOTE – When both alpha_channel_incr_flag and alpha_channel_clip_flag are equal to one, the clipping operation specified by alpha_channel_clip_type_flag should be applied first followed by the alteration specified by alpha_channel_incr_flag to obtain the interpretation sample value for the auxiliary picture luma sample.

F.14.3.9 Overlay information SEI message semantics

The overlay information SEI message provides information about overlay pictures coded as auxiliary pictures. Overlay auxiliary pictures have nuh_layer_id equal to nuhLayerIdA and AuxId[nuhLayerIdA] in the range of 128 to 159, inclusive. Each overlay auxiliary picture layer is associated with one or more primary picture layers as specified below.

overlay_info_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous overlay information SEI message in output order that is associated with one or more primary picture layers to which this SEI applies. overlay_info_cancel_flag equal to 0 indicates that overlay information follows.

overlay_content_aux_id_minus128 plus 128 indicates the value of AuxId of auxiliary pictures containing overlay content. overlay_content_aux_id_minus128 shall be in the range of 0 to 31, inclusive.

overlay_label_aux_id_minus128 plus 128 indicates the value of AuxId of auxiliary pictures containing overlay label. overlay_label_aux_id_minus128 shall be in the range of 0 to 31, inclusive.

overlay_alpha_aux_id_minus128 plus 128 indicates the value of AuxId of auxiliary pictures containing overlay alpha. overlay_alpha_aux_id_minus128 shall be in the range of 0 to 31, inclusive.

overlay_element_label_value_length_minus8 plus 8 indicates the number of bits used for coding the overlay_element_label_min[i][j] and overlay_element_label_max[i][j] syntax elements.

num_overlays_minus1 plus 1 specifies the number of overlays described by the overlay information SEI message. num_overlays_minus1 shall be in the range of 0 to 15, inclusive.

overlay_idx[i] indicates the index of the i-th overlay. overlay_idx[i] shall be in the range of 0 to 255, inclusive.

language_overlay_present_flag[i] equal to 1 indicates that overlay_language[i] is present. language_overlay_present_flag[i] equal to 0 indicates that overlay_language[i] is not present and that the language of the overlay is unspecified.

overlay_content_layer_id[i] indicates the nuh_layer_id value of the NAL units of the overlay content of the i-th overlay. AuxId[overlay_content_layer_id[i]] shall be equal to overlay_content_aux_id_minus128 + 128 for all values of i in the range of 0 to num_overlays_minus1, inclusive.

The value of the variable pLid, which identifies the nuh_layer_id value of the primary picture which the i-th overlay is associated with, is derived as follows:

```
pLid = -1
for( j = 0; j < 63; j++ )
    if( ViewOrderIdx[ j ] == ViewOrderIdx[ overlay_content_layer_id[ i ] ] &&
        DependencyId[ j ] == DependencyId[ overlay_content_layer_id[ i ] ] && AuxId[ j ] ==
0 )
        pLid = j
```

The value of pLid shall be in the range of 0 to 62, inclusive.

overlay_label_present_flag[i] equal to 1 specifies that overlay_label_layer_id[i] is present. overlay_label_present_flag[i] equal to 0 specifies that overlay_label_layer_id[i] is not present.

overlay_label_layer_id[i] indicates the nuh_layer_id value of NAL units in the overlay label of the i-th overlay. AuxId[overlay_label_layer_id[i]] shall be equal to overlay_label_aux_id_minus128 + 128 for all values of i in the range of 0 to num_overlays_minus1, inclusive.

overlay_alpha_present_flag[i] equal to 1 specifies that overlay_alpha_layer_id[i] is present. overlay_alpha_present_flag[i] equal to 0 specifies that overlay_alpha_layer_id[i] is not present.

overlay_alpha_layer_id[i] indicates the nuh_layer_id value of NAL units in the overlay alpha of the i-th overlay. AuxId[overlay_alpha_layer_id[i]] shall be equal to overlay_alpha_aux_id_minus128 + 128 for all values of i in the range of 0 to num_overlays_minus1, inclusive.

num_overlay_elements_minus1[i] indicates the number of overlay elements in the i-th overlay. num_overlay_elements_minus1[i] shall be in the range of 0 to 255, inclusive. When not present, the value of num_overlay_elements_minus1[i] is inferred to be equal to 0.

overlay_element_label_min[i][j] and **overlay_element_label_max**[i][j] indicate the minimum and maximum values, respectively, of the range of sample values corresponding to the j-th overlay element of the i-th overlay. The length of the **overlay_element_label_min**[i][j] and the **overlay_element_label_max**[i][j] syntax elements is **overlay_element_label_min_max_length_minus8** + 8 bits.

The variable **overlayElementId**[i][x][y] specifying the overlay element identifier of the i-th overlay for the sample location (x, y) relative to the top-left sample is derived as follows, where **p**[i][x][y] refers to the luma sample at location (x, y) in the decoded label auxiliary picture of the i-th overlay:

```

for( y = 0; y < pic_height_in_luma_samples; y++ )
  for( x = 0; x < pic_width_in_luma_samples; x++ )
    for( i = 0; i <= number_overlays_minus1[ i ] ) {
      overlayElementId[ i ][ x ][ y ] = 0
      for( j = 0; j <= num_overlay_elements_minus1[ i ]; j++ )
        if( p[ i ][ x ][ y ] >= overlay_element_label_min[ i ][ j ] &&
            p[ i ][ x ][ y ] <= overlay_element_label_max[ i ][ j ] )
          overlayElementId[ i ][ x ][ y ] = j
    }

```

(F-96)

overlay_zero_bit shall be equal to 0.

overlay_language[i] contains a language tag as specified by IETF RFC 5646 followed by a null termination byte equal to 0x00. The length of the **overlay_language**[i] syntax element shall be less than or equal to 255 bytes, not including the null termination byte.

overlay_name[i] indicates the name of the i-th overlay. The length of the **overlay_name**[i] syntax element shall be less than or equal to 255 bytes, not including the null termination byte.

overlay_element_name[i][j] indicates the name of the j-th overlay element of the i-th overlay. The length of the **overlay_element_name**[i][j] syntax element shall be less than or equal to 255 bytes, not including the null termination byte.

overlay_info_persistence_flag specifies the persistence of the overlay information SEI message. **overlay_info_persistence_flag** equal to 0 specifies that the overlay information SEI message applies to the current decoded picture only.

When an access unit contains an auxiliary picture **picA** with **nuh_layer_id** equal to **nuhLayerIdA** and **AuxId**[**nuhLayerIdA**] in the range of 128..159, and **nuhLayerIdA** is equal to any of **overlay_content_layer_id**[i], **overlay_label_layer_id**[i], **overlay_alpha_layer_id**[i], **overlay_info_persistence_flag** equal to 1 specifies that the overlay information SEI message persists for the CLVS containing the auxiliary picture **picA** in output order until one or more of the following conditions are true:

- A new CLVS begins for the layer containing the auxiliary picture **picA**.
- A new CLVS begins for the layer containing the primary picture associated with the auxiliary picture **picA**.
- The bitstream ends.
- A picture **picB** in an access unit containing an overlay information SEI message is output for which **PicOrderCnt**(**picB**) is greater than **PicOrderCnt**(**picA**), where **PicOrderCnt**(**picB**) and **PicOrderCnt**(**picA**) are the **PicOrderCntVal** values of **picB** and **picA**, respectively, immediately after the invocation of the decoding process for picture order count for **picB**.

F.14.3.10 Temporal motion vector prediction constraints SEI message semantics

The temporal motion vector prediction constraints SEI message indicates constraints on collocated pictures for temporal motion vector prediction. This SEI message may be used to determine whether the motion vectors of earlier pictures in decoding order no longer need to be stored and whether the motion vectors of the current picture and subsequent pictures need to be stored.

The temporal motion vector prediction constraints SEI message is a prefix SEI message. The temporal motion vector prediction constraints SEI message may be present in an access unit with **TemporalId** equal to 0 and shall not be present in an access unit with **TemporalId** greater than 0.

The following semantics apply separately to each **nuh_layer_id** **targetLayerId** among the **nuh_layer_id** values to which the temporal motion vector prediction constraints SEI message applies. Let **associatedLayerIdList** consist of each **targetLayerId** value to which this temporal motion vector prediction constraints SEI message applies.

Let a set of pictures **associatedPicSet** be the pictures with **nuh_layer_id** equal to **targetLayerId** from the access unit containing the SEI message, inclusive, up to but not including the first of any of the following in decoding order:

- The next access unit, in decoding order, that contains a temporal motion vector prediction constraints SEI message with an associatedLayerIdList that contains targetLayerId.
- The first picture of the next CLVS, in decoding order, of the layer with nuh_layer_id equal to targetLayerId.

prev_pics_not_used_flag equal to 1 indicates that the syntax elements for all coded pictures that are within or follow the access unit containing the current picture in decoding order are constrained such that no temporal motion vector from any picture that has nuh_layer_id equal to any value in associatedLayerIdList and precedes the access unit containing the current picture in decoding order is used directly or indirectly in decoding of any coded picture that is within or follows the access unit containing the current picture in decoding order. **prev_pics_not_used_flag** equal to 0 indicates that the bitstream may or may not fulfill the constraints indicated by **prev_pics_not_used_flag** equal to 1.

NOTE 1 – When **prev_pics_not_used_flag** is equal to 1, decoders may empty the "motion vector storage" for all reference pictures with nuh_layer_id equal to targetLayerId in the decoded picture buffer.

prev_pics_not_used_flag shall be equal to 1 when both of the following conditions are true:

- **no_intra_layer_col_pic_flag** is equal to 1 in the previous temporal motion vector prediction constraints SEI message applying to nuh_layer_id equal to targetLayerId.
- The previous temporal motion vector prediction constraints SEI message applying to nuh_layer_id equal to targetLayerId and the current temporal motion vector prediction constraints SEI message apply to the same CLVS of the layer with nuh_layer_id equal to targetLayerId.

no_intra_layer_col_pic_flag equal to 1 indicates the following:

- If NumDirectRefLayers[targetLayerId] is equal to 0, slice_temporal_mvp_enabled_flag is not present or is equal to 0 in each picture in associatedPicSet.
- Otherwise, all the pictures in associatedPicSet do not use temporal motion vector prediction or use collocated pictures with nuh_layer_id different from targetLayerId.

When **no_intra_layer_col_pic_flag** is equal to 0, no constraint on the collocated picture of the pictures with nuh_layer_id equal to targetLayerId is indicated.

Let NoIntraLayerColPicFlag[targetLayerId] be equal to **no_intra_layer_col_pic_flag**.

NOTE 2 – The motion vectors of the current picture with nuh_layer_id equal to layerId have to be stored when they may be used for temporal motion vector prediction of other pictures in the same layer or when they may be used for inter-layer motion prediction. In other words, the motion vectors of the current picture have to be stored when at least one of the following is true:

- **sps_temporal_mvp_enabled_flag** in the active SPS for the current picture is equal to 1 and NoIntraLayerColPicFlag[layerId] is equal 0.
- NoIntraLayerColPicFlag[layerId] is equal to 1 and there is a nuh_layer_id value nuhLayerIdA such that VpsInterLayerMotionPredictionEnabled[LayerIdxInVps[nuhLayerIdA]][LayerIdxInVps[layerId]] is equal to 1.

NOTE 3 – The motion vectors of a picture with nuh_layer_id equal to layerId need no longer be stored when the picture is marked as "unused for reference", or the picture is not used for temporal motion vector prediction of other pictures in the same layer and all pictures in the same access unit that may use the picture as a reference for inter-layer motion prediction have been decoded, or the access unit containing the picture precedes the current access unit in decoding order, where this SEI message is present with associatedLayerIdList including the nuh_layer_id of the picture and **prev_pics_not_used_flag** equal to 1.

F.14.3.11 Frame-field information SEI message semantics

The frame-field information SEI message may be used to indicate how the associated picture should be displayed, the source scan type of the associated picture, and whether the associated picture is a duplicate of a previous picture, in output order, of the same layer.

The following semantics apply separately to each nuh_layer_id targetLayerId among the nuh_layer_id values to which the frame-field information SEI message applies.

A frame-field information SEI message associated with nuh_layer_id equal to targetLayerId shall be present in an access unit, when all of the following conditions are true:

- A picture with nuh_layer_id equal to targetLayerId is present in the access unit.
- **frame_field_info_present_flag** is equal to 1 in the active SPS for the layer with nuh_layer_id equal to targetLayerId.
- targetLayerId is greater than 0 or none of the following conditions is true:
 - A non-scalable-nested picture timing SEI message is present in the access unit.
 - A picture timing SEI message directly contained in a scalable nesting SEI message is present in the access unit.

A frame-field information SEI message that applies to a particular set of layers shall not be present when one or more of the following conditions are true:

- The value of `field_seq_flag` is not the same for all active SPSs for the particular set of layers.
- The values of `general_progressive_source_flag` and `general_interlaced_source_flag` are not identical, respectively, for all the `profile_tier_level()` syntax structures that apply to the layers to which the frame-field information SEI message applies.

The semantics of `ffinfo_pic_struct`, `ffinfo_source_scan_type` and `ffinfo_duplicate_flag` apply layer-wise to each value of `targetLayerId`.

`ffinfo_pic_struct` has the same semantics as the `pic_struct` syntax element in the picture timing SEI message.

`ffinfo_source_scan_type` has the same semantics as the `source_scan_type` syntax element in the picture timing SEI message.

`ffinfo_duplicate_flag` has the same semantics as the `duplicate_flag` syntax element in the picture timing SEI message.

F.15 Video usability information

F.15.1 General

The specifications in clause E.1 apply.

F.15.2 VUI syntax

The specifications in clause E.2 and its subclauses apply.

F.15.3 VUI semantics

F.15.3.1 VUI parameters semantics

The specifications in clause E.3.1 apply by replacing the semantics of `field_seg_flag`, `frame_field_info_present_flag` and `vui_timing_info_present_flag` with those below and with the following additions:

- It is a requirement of bitstream conformance that, when `nuh_layer_id` `layerId` is greater than 0 and `NumDirectRefLayers[layerId]` is greater than 0, `video_signal_type_present_flag` present in or inferred for the active SPS for `nuh_layer_id` equal to `layerId` shall be equal to 0.
- When the current picture has `nuh_layer_id` `layerIdCurr` greater than 0, either `NumDirectRefLayers[layerIdCurr]` is greater than 0 or `MultiLayerExtSpsFlag` derived from the active SPS for the `nuh_layer_id` equal to `layerIdCurr` is equal to 1 and the active SPS for `nuh_layer_id` equal to `layerIdCurr` contains the VUI parameters syntax structure, the following applies:
 - The values of `video_format`, `video_full_range_flag`, `colour_primaries`, `transfer_characteristics` and `matrix_coeffs` are inferred to be equal to `video_vps_format`, `video_full_range_vps_flag`, `colour_primaries_vps`, `transfer_characteristics_vps` and `matrix_coeffs_vps`, respectively, of the `vps_video_signal_info_idx[j]`-th `video_signal_info()` syntax structure in the active VPS where `j` is equal to `LayerIdxInVps[layerIdCurr]`.
 - The values of `video_format`, `video_full_range_flag`, `colour_primaries`, `transfer_characteristics` and `matrix_coeffs` signalled in the active SPS for the layer with `nuh_layer_id` equal to `layerIdCurr` are ignored.

NOTE 1 – The values are inferred from the VPS when a non-base layer refers to an SPS that is also referred to by the base layer, in which case the SPS has `nuh_layer_id` equal to 0. For the base layer, the values of these parameters in the active SPS for the base layer apply.

`field_seq_flag` equal to 1 indicates that the layers for which the SPS is an active SPS within the CVS convey pictures that represent fields and specifies the following:

- When the SPS is an active SPS for `nuh_layer_id` equal to 0, a picture timing SEI message that is not scalable-nested or that is directly contained in a scalable nesting SEI message and applies to `nuh_layer_id` equal to 0 shall be present in every such access unit of the current CVS that contains a coded picture with `nuh_layer_id` equal to 0.
- When the SPS is an active SPS for the `nuh_layer_id` value `nuhLayerId` greater than 0, a frame-field information SEI message shall be present for `nuh_layer_id` equal to `nuhLayerId` in every access unit containing a picture for the current CLVS of the layer with `nuh_layer_id` equal to `nuhLayerId`.

`field_seq_flag` equal to 0 indicates that the layers for which the SPS is an active SPS within the CVS convey pictures that represent frames and that a picture timing SEI message or a frame-field information SEI message may or may not be present for the pictures within the CVS belonging to any layer for which the SPS is an active SPS. When `field_seq_flag` is not present, it is inferred to be equal to 0. When `general_frame_only_constraint_flag` is present in the SPS and is equal to 1, the value of `field_seq_flag` shall be equal to 0. When `general_frame_only_constraint_flag` is present in the active VPS, applies for a layer for which the SPS is an active SPS and is equal to 1, the value of `field_seq_flag` shall be equal to 0.

NOTE 2 – The specified decoding process does not treat access units conveying pictures that represent fields or frames differently. A sequence of pictures that represent fields would therefore be coded with the picture dimensions of an individual field. For example, access units containing pictures that represent 1080i fields would commonly have cropped output dimensions of 1920x540, while the sequence picture rate would commonly express the rate of the source fields (typically between 50 and 60 Hz), instead of the source frame rate (typically between 25 and 30 Hz).

frame_field_info_present_flag equal to 1 specifies that picture timing SEI messages or frame-field information SEI messages are present for every picture for which this SPS is the active SPS and the picture timing SEI messages, when present, include the `pic_struct`, `source_scan_type`, and `duplicate_flag` syntax elements. **frame_field_info_present_flag** equal to 0 specifies that the `pic_struct` syntax element is not present in picture timing SEI messages associated with pictures for which the SPS is the active SPS.

When `frame_field_info_present_flag` is present and either or both of the following conditions are true, `frame_field_info_present_flag` shall be equal to 1:

- `field_seq_flag` is equal to 1.
- `general_progressive_source_flag` and `general_interlaced_source_flag` are present in this SPS, `general_progressive_source_flag` is equal to 1 and `general_interlaced_source_flag` is equal to 1.

When `frame_field_info_present_flag` is not present, its value is inferred as follows:

- If `general_progressive_source_flag` and `general_interlaced_source_flag` are present in this SPS, `general_progressive_source_flag` is equal to 1 and `general_interlaced_source_flag` is equal to 1, `frame_field_info_present_flag` is inferred to be equal to 1.
- Otherwise, `frame_field_info_present_flag` is inferred to be equal to 0.

vui_timing_info_present_flag equal to 1 specifies that `vui_num_units_in_tick`, `vui_time_scale`, `vui_poc_proportional_to_timing_flag` and `vui_hrd_parameters_present_flag` are present in the `vui_parameters()` syntax structure. **vui_timing_info_present_flag** equal to 0 specifies that `vui_num_units_in_tick`, `vui_time_scale`, `vui_poc_proportional_to_timing_flag` and `vui_hrd_parameters_present_flag` are not present in the `vui_parameters()` syntax structure. It is a requirement of bitstream conformance that, when `MultiLayerExtSpsFlag` is equal to 1, `vui_timing_info_present_flag` shall be equal to 0.

F.15.3.2 HRD parameters semantics

The specifications in clause E.3.2 apply with the replacement of each instance of "access unit level" and "sub-picture level" with "partition unit level" and "sub-partition level", respectively, and with the following additions:

`initial_cpb_removal_delay_length_minus1` plus 1 within the *j*-th `hrd_parameters()` syntax structure in the VPS specifies the length, in bits, of the `nal_initial_arrival_delay[k]` and `vcl_initial_arrival_delay[k]` syntax elements of the bitstream partition initial arrival time SEI message that is contained in a bitstream partition nesting SEI message within a scalable nesting SEI message with values of `MaxTemporalId[0]`, `sei_ols_idx`, `sei_partitioning_scheme_idx` and `bsp_idx` such that `bsp_hrd_idx[sei_ols_idx][sei_partitioning_scheme_idx][MaxTemporalId[0]][k][bsp_idx]` is equal to *j*. When the `initial_cpb_removal_delay_length_minus1` syntax element is not present, it is inferred to be equal to 23.

It is a requirement of bitstream conformance that the value of `sub_pic_hrd_params_present_flag` shall be the same for all `hrd_parameters()` syntax structures that apply to at least one bitstream partition of a particular output layer set.

F.15.3.3 Sub-layer HRD parameters semantics

The specifications in clause E.3.3 apply.

Annex G

Multiview high efficiency video coding

(This annex forms an integral part of this Recommendation | International Standard.)

G.1 Scope

This annex specifies syntax, semantics and decoding processes for multiview high efficiency video coding that use the syntax, semantics and decoding processes specified in clauses 2-10 and Annexes A-F. This annex also specifies profiles, tiers and levels for multiview high efficiency video coding.

G.2 Normative references

The list of normative references in clause F.2 applies.

G.3 Definitions

The specifications in clause F.3 and its subclauses apply.

G.4 Abbreviations

The specifications in clause F.4 apply.

G.5 Conventions

The specifications in clause F.5 apply.

G.6 Bitstream and picture formats, partitionings, scanning processes, and neighbouring relationships

The specifications in clause F.6 apply.

G.7 Syntax and semantics

The specifications in clause F.7 and its subclauses apply.

G.8 Decoding processes

G.8.1 General decoding process

G.8.1.1 General

The specifications of clause F.8.1.1 apply.

G.8.1.2 Decoding process for a coded picture with `nuh_layer_id` greater than 0

The decoding process for the current picture CurrPic is as follows:

1. The decoding of NAL units is specified in clause G.8.2.
2. The processes in clauses G.8.1.3 and F.8.3.4 specify the following decoding processes using syntax elements in the slice segment layer and above:
 - Prior to decoding the first slice of the current picture, clause G.8.1.3 is invoked.
 - At the beginning of the decoding process for each P or B slice, the decoding process for reference picture lists construction specified in clause F.8.3.4 is invoked for derivation of reference picture list 0 (RefPicList0), and when decoding a B slice, reference picture list 1 (RefPicList1).
3. The processes in clauses G.8.4, G.8.5, G.8.6 and G.8.7 specify decoding processes using syntax elements in all syntax structure layers. It is a requirement of bitstream conformance that the coded slices of the picture shall contain slice segment data for every CTU of the picture, such that the division of the picture into slices, the division of the slices into slice segments and the division of the slice segments into CTUs each form a partitioning of the picture.

G.8.1.3 Decoding process for inter-layer reference picture set

Outputs of this process are updated lists of inter-layer reference pictures RefPicSetInterLayer0 and RefPicSetInterLayer1 and the variables NumActiveRefLayerPics0 and NumActiveRefLayerPics1.

The variable currLayerId is set equal to nuh_layer_id of the current picture.

The lists RefPicSetInterLayer0 and RefPicSetInterLayer1 are first emptied, NumActiveRefLayerPics0 and NumActiveRefLayerPics1 are set equal to 0 and the following applies:

```
for( i = 0; i < NumActiveRefLayerPics; i++ ) {
    refPicSet0Flag =
        ( ( ViewId[ currLayerId ] <= ViewId[ 0 ] && ViewId[ currLayerId ] <=
ViewId[ RefPicLayerId[ i ] ] ) ||
        ( ViewId[ currLayerId ] >= ViewId[ 0 ] && ViewId[ currLayerId ] >=
ViewId[ RefPicLayerId[ i ] ] ) )
    if( there is a picture picX in the DPB that is in the same access unit as the current picture and has
        nuh_layer_id equal to RefPicLayerId[ i ] ) {
        if( refPicSet0Flag ) {
            RefPicSetInterLayer0[ NumActiveRefLayerPics0 ] = picX
            RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] is marked as "used for long-term
reference"
        } else {
            RefPicSetInterLayer1[ NumActiveRefLayerPics1 ] = picX
            RefPicSetInterLayer1[ NumActiveRefLayerPics1++ ] is marked as "used for long-term
reference"
        }
    } else {
        if( refPicSet0Flag )
            RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] = "no reference picture"
        else
            RefPicSetInterLayer1[ NumActiveRefLayerPics1++ ] = "no reference picture"
    }
}
```

(G-1)

There shall be no entry equal to "no reference picture" in RefPicSetInterLayer0 or RefPicSetInterLayer1.

There shall be no picture that has discardable_flag equal to 1 in RefPicSetInterLayer0 or RefPicSetInterLayer1.

If the current picture is a RADL picture, there shall be no entry in RefPicSetInterLayer0 or RefPicSetInterLayer1 that is a RASL picture.

NOTE – An access unit may contain both RASL and RADL pictures.

G.8.2 NAL unit decoding process

The specifications in clause F.8.2 apply.

G.8.3 Slice decoding processes

The specifications in clause F.8.3 and its subclauses apply.

G.8.4 Decoding process for coding units coded in intra prediction mode

The specifications in clause F.8.4 apply.

G.8.5 Decoding process for coding units coded in inter prediction mode

The specifications in clause F.8.5 apply.

G.8.6 Scaling, transformation and array construction process prior to deblocking filter process

The specifications in clause F.8.6 apply.

G.8.7 In-loop filter process

The specifications in clause F.8.7 apply.

G.9 Parsing process

The specifications in clause F.9 apply.

G.10 Specification of bitstream subsets

The specifications in clause F.10 and its subclauses apply.

G.11 Profiles, tiers and levels

G.11.1 Profiles

G.11.1.1 Multiview Main profile

For a layer in an output operation point associated with an OLS in a bitstream, the layer being conforming to the Multiview Main profile, the following applies:

- Let `olsIdx` be the OLS index of the OLS, the sub-bitstream `subBitstream` and the base layer sub-bitstream `baseBitstream` are derived as specified in clause F.11.3.

When `vps_base_layer_internal_flag` is equal to 1, the base layer sub-bitstream `baseBitstream` shall obey the following constraints:

- The base layer sub-bitstream `baseBitstream` shall be indicated to conform to the Main profile.

The sub-bitstream `subBitstream` shall obey the following constraints:

- All active VPSs shall have `vps_num_rep_formats_minus1` in the range of 0 to 15, inclusive.
- All active SPSs for layers in `subBitstream` shall have `chroma_format_idc` equal to 1 only.
- All active SPSs for layers in `subBitstream` shall have `transform_skip_rotation_enabled_flag`, `transform_skip_context_enabled_flag`, `implicit_rdpkm_enabled_flag`, `explicit_rdpkm_enabled_flag`, `extended_precision_processing_flag`, `intra_smoothing_disabled_flag`, `high_precision_offsets_enabled_flag`, `persistent_rice_adaptation_enabled_flag` and `cabac_bypass_alignment_enabled_flag`, when present, equal to 0 only.
- `CtbLog2SizeY` derived from all active SPSs for layers in `subBitstream` shall be in the range of 4 to 6, inclusive.
- All active PPSs for layers in `subBitstream` shall have `log2_max_transform_skip_block_size_minus2` and `chroma_qp_offset_list_enabled_flag`, when present, equal to 0 only.
- `ScalabilityId[j][smIdx]` derived according to any active VPS shall be equal to 0 for any `smIdx` value not equal to 1 or 3 and for any value of `j` such that `layer_id_in_nuh[j]` is among `layerIdListTarget` that was used to derive `subBitstream`.
- When `NumLayersInIdList[OlsIdxToLsIdx[olsIdx]]` is equal to 2, `output_layer_flag[olsIdx][j]` derived according to any active VPS shall be equal to 1 for `j` in the range of 0 to 1, inclusive, for `subBitstream`.
- All active VPSs shall have `alt_output_layer_flag[olsIdx]` equal to 0 only.
- When `ViewOrderIdx[i]` derived according to any active VPS is equal to 1 for the layer with `nuh_layer_id` equal to `i` in `subBitstream`, `inter_view_mv_vert_constraint_flag` shall be equal to 1 in the `sps_multilayer_extension()` syntax structure in each active SPS for that layer.
- When `ViewOrderIdx[i]` derived according to any active VPS is greater than 0 for the layer with `nuh_layer_id` equal to `i` in `subBitstream`, `num_ref_loc_offsets` shall be equal to 0 in each active PPS for that layer.
- When `ViewOrderIdx[i]` derived according to any active VPS is greater than 0 for the layer with `nuh_layer_id` equal to `i` in `subBitstream`, the values of `pic_width_in_luma_samples` and `pic_height_in_luma_samples` in each active SPS for that layer shall be equal to the values of `pic_width_in_luma_samples` and `pic_height_in_luma_samples`, respectively, in each active SPS for all reference layers of that layer.
- For a layer with `nuh_layer_id` `iNuhLid` equal to any value included in `layerIdListTarget` that was used to derive `subBitstream`, the value of `NumRefLayers[iNuhLid]`, which specifies the total number of direct and indirect reference layers and is derived as specified in F.7.4.3.1, shall be less than or equal to 4.
- All active SPSs for layers in `subBitstream` shall have `sps_range_extension_flag` and `sps_scc_extension_flag` equal to 0 only.
- All active PPSs for layers in `subBitstream` shall have `pps_range_extension_flag` and `pps_scc_extension_flag` equal to 0 only.

- All active SPSs for layers in subBitstream shall have `bit_depth_luma_minus8` equal to 0 only.
- All active SPSs for layers in subBitstream shall have `bit_depth_chroma_minus8` equal to 0 only.
- All active PPSs for layers in subBitstream shall have `colour_mapping_enabled_flag` equal to 0 only.
- When an active PPS for any layer in subBitstream has `tiles_enabled_flag` equal to 1, it shall have `entropy_coding_sync_enabled_flag` equal to 0.
- When an active PPS for any layer in subBitstream has `tiles_enabled_flag` equal to 1, `ColumnWidthInLumaSamples[i]` shall be greater than or equal to 256 for all values of `i` in the range of 0 to `num_tile_columns_minus1`, inclusive and `RowHeightInLumaSamples[j]` shall be greater than or equal to 64 for all values of `j` in the range of 0 to `num_tile_rows_minus1`, inclusive.
- The number of times `read_bits(1)` is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing `coding_tree_unit()` data for any CTU shall be less than or equal to $5 * \text{RawCtuBits} / 3$.
- `general_level_idc` and `sub_layer_level_idc[i]` for all values of `i` in active SPSs for any layer in subBitstream shall not be equal to 255 (which indicates level 8.5).
- The tier and level constraints specified for the Multiview Main profile in clause G.11.2 shall be fulfilled.
- For any active VPS, `ViewOrderIdx[i]` shall be greater than `ViewOrderIdx[j]` for any values of `i` and `j` among `layerIdListTarget` that was used to derive subBitstream such that `AuxIdx[i]` is equal to `AuxIdx[j]` and `i` is greater than `j`.

In the remainder of this clause and clause G.11.2.1, all syntax elements in the `profile_tier_level()` syntax structure refer to those in the `profile_tier_level()` syntax structure associated with the layer.

Conformance of a layer in an output operation point associated with an OLS in a bitstream to the Multiview Main profile is indicated as follows:

- If `OpTid` of the output operation point is equal to `vps_max_sub_layer_minus1`, the conformance is indicated by `general_profile_idc` being equal to 6 or `general_profile_compatibility_flag[6]` being equal to 1 and `general_max_12bit_constraint_flag` being equal to 1, `general_max_10bit_constraint_flag` being equal to 1, `general_max_8bit_constraint_flag` being equal to 1, `general_max_422chroma_constraint_flag` being equal to 1, `general_max_420chroma_constraint_flag` being equal to 1, `general_max_monochrome_constraint_flag` being equal to 0, `general_intra_constraint_flag` being equal to 0 and `general_one_picture_only_constraint_flag` being equal to 0 and `general_lower_bit_rate_constraint_flag` being equal to 1.
- Otherwise (`OpTid` of the output operation point is less than `vps_max_sub_layer_minus1`), the conformance is indicated by `sub_layer_profile_idc[OpTid]` being equal to 6 or `sub_layer_profile_compatibility_flag[OpTid][6]` being equal to 1 and `sub_layer_max_12bit_constraint_flag[OpTid]` being equal to 1, `sub_layer_max_10bit_constraint_flag[OpTid]` being equal to 1, `sub_layer_max_8bit_constraint_flag[OpTid]` being equal to 1, `sub_layer_max_422chroma_constraint_flag[OpTid]` being equal to 1, `sub_layer_max_420chroma_constraint_flag[OpTid]` being equal to 1, `sub_layer_max_monochrome_constraint_flag[OpTid]` being equal to 0, `sub_layer_intra_constraint_flag[OpTid]` being equal to 0 and `sub_layer_one_picture_only_constraint_flag[OpTid]` being equal to 0 and `sub_layer_lower_bit_rate_constraint_flag[OpTid]` being equal to 1.

G.11.2 Tiers and levels

G.11.2.1 General tier and level limits

For purposes of comparison of tier capabilities, the tier with `general_tier_flag` or `sub_layer_tier_flag[i]` equal to 0 is considered to be a lower tier than the tier with `general_tier_flag` or `sub_layer_tier_flag[i]` equal to 1.

For purposes of comparison of level capabilities, a particular level of a specific tier is considered to be a lower level than some other level of the same tier when the value of the `general_level_idc` or `sub_layer_level_idc[i]` of the particular level is less than that of the other level.

The following is specified for expressing the constraints in this clause and clause G.11.2.2:

- The value of each of the variables `CpbVclFactor`, `CpbNalFactor`, `FormatCapabilityFactor` and `MinCrScaleFactor` is the same as that specified in Table A.10 for the Main profile.
- Let access unit `n` be the `n`-th access unit in decoding order, with the first access unit being access unit 0 (i.e., the 0-th access unit).
- Let the variable `fR` be set equal to $1 \div 300$.
- Let the variable `olsIdx` be the index of the OLS.

- For each layer with `nuh_layer_id` equal to `currLayerId`, let the variable `layerSizeInSamplesY` be derived as follows:

$$\text{layerSizeInSamplesY} = \text{pic_width_vps_in_luma_samples} * \text{pic_height_vps_in_luma_samples} \quad (\text{G-2})$$

where `pic_width_vps_in_luma_samples` and `pic_height_vps_in_luma_samples` are found in the `vps_rep_format_idx[LayerIdxInVps[currLayerId]]-th rep_format()` syntax structure in the VPS.

Each layer with `nuh_layer_id` equal to `currLayerId` conforming to a profile at a specified tier and level shall obey the following constraints for each bitstream conformance test as specified in clause F.13, where "access unit" is used to denote the picture unit in the layer and the CPB is understood to be the BPB:

- The value of `layerSizeInSamplesY` shall be less than or equal to `MaxLumaPs`, where `MaxLumaPs` is specified in Table A.8 for the tier and level of the layer.
- The value of `pic_width_vps_in_luma_samples` of the `vps_rep_format_idx[LayerIdxInVps[currLayerId]]-th rep_format()` syntax structure in the VPS shall be less than or equal to $\text{Sqrt}(\text{MaxLumaPs} * 8)$.
- The value of `pic_height_vps_in_luma_samples` of the `vps_rep_format_idx[LayerIdxInVps[currLayerId]]-th rep_format()` syntax structure in the VPS shall be less than or equal to $\text{Sqrt}(\text{MaxLumaPs} * 8)$.
- The value of `max_vps_dec_pic_buffering_minus1[olsIdx][LayerIdxInVps[currLayerId]][HighestTid]` shall be less than or equal to `MaxDpbSize` as derived by Equation A-2, with `PicSizeInSamplesY` being replaced with `layerSizeInSamplesY`, for the tier and level of the layer.
- For level 5 and higher levels, the value of `CtbSizeY` for the layer shall be equal to 32 or 64.
- The value of `NumPicTotalCurr` for each picture in the layer shall be less than or equal to 8.
- When decoding each coded picture in the layer, the value of `num_tile_columns_minus1` shall be less than `MaxTileCols` and `num_tile_rows_minus1` shall be less than `MaxTileRows`, where `MaxTileCols` and `MaxTileRows` are specified in Table A.8 for the tier and level of the layer.
- For the VCL HRD parameters of the layer, `CpbSize[i]` shall be less than or equal to `CpbVclFactor * MaxCPB` for at least one of the delivery schedules identified by `bsp_sched_idx[olsIdx][0][HighestTid][combIdx][LayerIdxInVps[currLayerId]]` for `combIdx` ranging from 0 to `num_bsp_schedules_minus1[olsIdx][0][HighestTid]`, inclusive, where `CpbSize[i]` is specified in clause F.13.1 and `MaxCPB` is specified in Table A.8 for the tier and level of the layer in units of `CpbVclFactor` bits.
- For the NAL HRD parameters of the layer, `CpbSize[i]` shall be less than or equal to `CpbNalFactor * MaxCPB` for at least one of the delivery schedules identified by `bsp_sched_idx[olsIdx][0][HighestTid][combIdx][LayerIdxInVps[currLayerId]]` for `combIdx` ranging from 0 to `num_bsp_schedules_minus1[olsIdx][0][HighestTid]`, inclusive, where `CpbSize[i]` is specified in clause F.13.1 and `MaxCPB` is specified in Table A.8 for the tier and level of the layer in units of `CpbNalFactor` bits.

Table A.8 specifies the limits for each level of each tier for levels other than level 8.5.

A tier and level to which a layer in an output operation point associated with an OLS in a bitstream conforms are indicated by the syntax elements `general_tier_flag` and `general_level_idc` if `OpTid` of the output layer set is equal to `vps_max_sub_layer_minus1`, and by the syntax elements `sub_layer_tier_flag[OpTid]` and `sub_layer_level_idc[OpTid]` otherwise, as follows:

- If the specified level is not level 8.5, `general_tier_flag` or `sub_layer_tier_flag[OpTid]` equal to 0 indicates conformance to the Main tier, and `general_tier_flag` or `sub_layer_tier_flag[OpTid]` equal to 1 indicates conformance to the High tier, according to the tier constraints specified in Table A.8, and `general_tier_flag` and `sub_layer_tier_flag[OpTid]` shall be equal to 0 for levels below level 4 (corresponding to the entries in Table A.8 marked with "-"). Otherwise (the specified level is level 8.5), it is a requirement of bitstream conformance that `general_tier_flag` and `sub_layer_tier_flag[OpTid]` shall be equal to 1 and the value 0 for `general_tier_flag` and `sub_layer_tier_flag[OpTid]` is reserved for future use by ITU-T | ISO/IEC and decoders shall ignore the value of `general_tier_flag` and `sub_layer_tier_flag[OpTid]`.
- `general_level_idc` and `sub_layer_level_idc[OpTid]` shall be set equal to a value of 30 times the level number specified in Table A.8.

G.11.2.2 Profile-specific tier and level limits for the Multiview Main profile

The following is specified for expressing the constraints in this clause:

- The variable `HbrFactor` is set equal to 1.

- The variable BrVclFactor is set equal to $CpbVclFactor * HbrFactor$.
- The variable BrNalFactor is set equal to $CpbNalFactor * HbrFactor$.
- The variable MinCr is set equal to $MinCrBase * MinCrScaleFactor \div HbrFactor$, where MinCrBase is specified in Table A.9.

Each layer conforming to the Multiview Main profile at a specified tier and level shall obey the following constraints for each conformance test as specified in clause F.13, where "access unit" is used to denote the picture unit in the layer, and the CPB is understood to be the BPB:

- a) The nominal removal time of access unit n (with n greater than 0) from the CPB, as specified in clause F.13.2.3, shall satisfy the constraint that $AuNominalRemovalTime[n] - AuCpbRemovalTime[n - 1]$ is greater than or equal to $Max(layerSizeInSamplesY \div MaxLumaSr, fR)$, where $layerSizeInSamplesY$ is the value of $layerSizeInSamplesY$ for access unit $n - 1$ and $MaxLumaSr$ is the value specified in Table A.9 that applies to access unit $n - 1$ for the tier and level of the layer.
- b) The difference between consecutive output times of pictures in different access units, as specified in clause F.13.3.3, shall satisfy the constraint that $DpbOutputInterval[n]$ is greater than or equal to $Max(layerSizeInSamplesY \div MaxLumaSr, fR)$, where $layerSizeInSamplesY$ is the value of $layerSizeInSamplesY$ of access unit n and $MaxLumaSr$ is the value specified in Table A.9 for access unit n for the tier and level of the layer, provided that access unit n is an access unit that has a picture that is output and is not the last of such access units.
- c) The removal time of access unit 0 shall satisfy the constraint that the number of coded slice segments in access unit 0 is less than or equal to $Min(Max(1, MaxSliceSegmentsPerPicture * MaxLumaSr / MaxLumaPs * (AuCpbRemovalTime[0] - AuNominalRemovalTime[0]) + MaxSliceSegmentsPerPicture * layerSizeInSamplesY / MaxLumaPs), MaxSliceSegmentsPerPicture)$, for the value of $layerSizeInSamplesY$ of access unit 0, where $MaxSliceSegmentsPerPicture$, $MaxLumaPs$ and $MaxLumaSr$ are the values specified in Table A.8 and Table A.9 for the tier and level of the layer.
- d) The difference between consecutive CPB removal times of access units n and $n - 1$ (with n greater than 0) shall satisfy the constraint that the number of slice segments in access unit n is less than or equal to $Min(Max(1, MaxSliceSegmentsPerPicture * MaxLumaSr / MaxLumaPs * (AuCpbRemovalTime[n] - AuCpbRemovalTime[n - 1])), MaxSliceSegmentsPerPicture)$, where $MaxSliceSegmentsPerPicture$, $MaxLumaPs$ and $MaxLumaSr$ are the values specified in Table A.8 and Table A.9 that apply to access unit n for the tier and level of the layer.
- e) For the VCL HRD parameters for the layer, $BitRate[i]$ shall be less than or equal to $BrVclFactor * MaxBR$ for at least one of the delivery schedules identified by $bsp_sched_idx[olsIdx][0][HighestTid][combIdx][LayerIdxInVps[currLayerId]]$ for $combIdx$ ranging from 0 to $num_bsp_schedules_minus1[olsIdx][0][HighestTid]$, inclusive, where $BitRate[i]$ is specified in clause F.13.1 and $MaxBR$ is specified in Table A.9 in units of $BrVclFactor$ bits/s for the tier and level of the layer.
- f) For the NAL HRD parameters for the layer, $BitRate[i]$ shall be less than or equal to $BrNalFactor * MaxBR$ for at least one of the delivery schedules identified by $bsp_sched_idx[olsIdx][0][HighestTid][combIdx][LayerIdxInVps[currLayerId]]$ for $combIdx$ ranging from 0 to $num_bsp_schedules_minus1[olsIdx][0][HighestTid]$, inclusive, where $BitRate[i]$ is specified in clause F.13.1 and $MaxBR$ is specified in Table A.9 in units of $BrNalFactor$ bits/s for the tier and level of the layer.
- g) The sum of the $NumBytesInNalUnit$ variables for access unit 0 shall be less than or equal to $FormatCapabilityFactor * (Max(layerSizeInSamplesY, fR * MaxLumaSr) + MaxLumaSr * (AuCpbRemovalTime[0] - AuNominalRemovalTime[0])) \div MinCr$ for the value of $layerSizeInSamplesY$ of access unit 0, where $MaxLumaSr$ is specified in Table A.9, and both $MaxLumaSr$ and $FormatCapabilityFactor$ are the values that apply to access unit 0 for the tier and level of the layer.
- h) The sum of the $NumBytesInNalUnit$ variables for access unit n (with n greater than 0) shall be less than or equal to $FormatCapabilityFactor * MaxLumaSr * (AuCpbRemovalTime[n] - AuCpbRemovalTime[n - 1]) \div MinCr$, where $MaxLumaSr$ is specified in Table A.9, and both $MaxLumaSr$ and $FormatCapabilityFactor$ are the values that apply to access unit n for the tier and level of the layer.
- i) The removal time of access unit 0 shall satisfy the constraint that the number of tiles in coded pictures in access unit 0 is less than or equal to $Min(Max(1, MaxTileCols * MaxTileRows * 120 * (AuCpbRemovalTime[0] - AuNominalRemovalTime[0]) + MaxTileCols * MaxTileRows * PicSizeInSamplesY / MaxLumaPs), MaxTileCols * MaxTileRows)$, for the value of $layerSizeInSamplesY$ of access unit 0, where $MaxTileCols$ and $MaxTileRows$ are the values specified in Table A.8 that apply to access unit 0 for the tier and level of the layer.

- j) The difference between consecutive CPB removal times of access units n and $n - 1$ (with n greater than 0) shall satisfy the constraint that the number of tiles in coded pictures in access unit n is less than or equal to $\text{Min}(\text{Max}(1, \text{MaxTileCols} * \text{MaxTileRows} * 120 * (\text{AuCpbRemovalTime}[n] - \text{AuCpbRemovalTime}[n - 1])), \text{MaxTileCols} * \text{MaxTileRows})$, where MaxTileCols and MaxTileRows are the values specified in Table A.8 that apply to access unit n for the tier and level of the layer.

G.11.3 Decoder capabilities

When a decoder conforms to any profile specified in Annex G, it shall also have the INBLD capability specified in clause F.11.1.

Clause F.11.2 specifies requirements for a decoder conforming to any profile specified in Annex G.

G.12 Byte stream format

The specifications in clause F.12 apply.

G.13 Hypothetical reference decoder

The specifications in clause F.13 and its subclauses apply.

G.14 Supplemental enhancement information

G.14.1 General

The specifications in clause F.14.1 apply.

G.14.2 SEI payload syntax

G.14.2.1 General SEI payload syntax

The specifications in clause F.14.2.1 apply.

G.14.2.2 Annex D and Annex F SEI message syntax for multiview high efficiency video coding

The specifications in clauses F.14.2.2 through F.14.2.11 apply.

G.14.2.33D reference displays information SEI message syntax

	Descriptor
three_dimensional_reference_displays_info (payloadSize) {	
prec_ref_display_width	ue(v)
ref_viewing_distance_flag	u(1)
if(ref_viewing_distance_flag)	
prec_ref_viewing_dist	ue(v)
num_ref_displays_minus1	ue(v)
for(i = 0; i <= num_ref_displays_minus1; i++) {	
left_view_id [i]	ue(v)
right_view_id [i]	ue(v)
exponent_ref_display_width [i]	u(6)
mantissa_ref_display_width [i]	u(v)
if(ref_viewing_distance_flag) {	
exponent_ref_viewing_distance [i]	u(6)
mantissa_ref_viewing_distance [i]	u(v)
}	
additional_shift_present_flag [i]	u(1)
if(additional_shift_present_flag[i])	
num_sample_shift_plus512 [i]	u(10)
}	
three_dimensional_reference_displays_extension_flag	u(1)
}	

G.14.2.4 Depth representation information SEI message syntax

G.14.2.4.1 General

depth_representation_info(payloadSize) {	Descriptor
z_near_flag	u(1)
z_far_flag	u(1)
d_min_flag	u(1)
d_max_flag	u(1)
depth_representation_type	ue(v)
if(d_min_flag d_max_flag)	
disparity_ref_view_id	ue(v)
if(z_near_flag)	
depth_rep_info_element(ZNearSign, ZNearExp, ZNearMantissa, ZNearManLen)	
if(z_far_flag)	
depth_rep_info_element(ZFarSign, ZFarExp, ZFarMantissa, ZFarManLen)	
if(d_min_flag)	
depth_rep_info_element(DMinSign, DMinExp, DMinMantissa, DMinManLen)	
if(d_max_flag)	
depth_rep_info_element(DMaxSign, DMaxExp, DMaxMantissa, DMaxManLen)	
if(depth_representation_type == 3) {	
depth_nonlinear_representation_num_minus1	ue(v)
for(i = 1; i <= depth_nonlinear_representation_num_minus1 + 1; i++)	
depth_nonlinear_representation_model[i]	ue(v)
}	
}	

G.14.2.4.2 Depth representation information element syntax

depth_rep_info_element(OutSign, OutExp, OutMantissa, OutManLen) {	Descriptor
da_sign_flag	u(1)
da_exponent	u(7)
da_mantissa_len_minus1	u(5)
da_mantissa	u(v)
}	

G.14.2.5 Multiview scene information SEI message syntax

multiview_scene_info(payloadSize) {	Descriptor
min_disparity	se(v)
max_disparity_range	ue(v)
}	

G.14.2.6 Multiview acquisition information SEI message syntax

	Descriptor
multiview_acquisition_info (payloadSize) {	
intrinsic_param_flag	u(1)
extrinsic_param_flag	u(1)
if(intrinsic_param_flag) {	
intrinsic_params_equal_flag	u(1)
prec_focal_length	ue(v)
prec_principal_point	ue(v)
prec_skew_factor	ue(v)
for(i = 0; i <= intrinsic_params_equal_flag ? 0 : numViewsMinus1; i++) {	
sign_focal_length_x [i]	u(1)
exponent_focal_length_x [i]	u(6)
mantissa_focal_length_x [i]	u(v)
sign_focal_length_y [i]	u(1)
exponent_focal_length_y [i]	u(6)
mantissa_focal_length_y [i]	u(v)
sign_principal_point_x [i]	u(1)
exponent_principal_point_x [i]	u(6)
mantissa_principal_point_x [i]	u(v)
sign_principal_point_y [i]	u(1)
exponent_principal_point_y [i]	u(6)
mantissa_principal_point_y [i]	u(v)
sign_skew_factor [i]	u(1)
exponent_skew_factor [i]	u(6)
mantissa_skew_factor [i]	u(v)
}	
}	
if(extrinsic_param_flag) {	
prec_rotation_param	ue(v)
prec_translation_param	ue(v)
for(i = 0; i <= numViewsMinus1; i++)	
for(j = 0; j < 3; j++) { /* row */	
for(k = 0; k < 3; k++) { /* column */	
sign_r [i][j][k]	u(1)
exponent_r [i][j][k]	u(6)
mantissa_r [i][j][k]	u(v)
}	
sign_t [i][j]	u(1)
exponent_t [i][j]	u(6)
mantissa_t [i][j]	u(v)
}	
}	
}	

G.14.2.7 Multiview view position SEI message syntax

<code>multiview_view_position(payloadSize) {</code>	Descriptor
<code> num_views_minus1</code>	<code>ue(v)</code>
<code> for(i = 0; i <= num_views_minus1; i++)</code>	
<code> view_position[i]</code>	<code>ue(v)</code>
<code>}</code>	

G.14.3 SEI payload semantics

G.14.3.1 General SEI payload semantics

The specifications in clause F.14.3.1 apply with the following modifications and additions:

The list `VclAssociatedSeiList` is set to consist of the `payloadType` values 2, 3, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 131, 132, 134 to 152, inclusive, 154 to 159, inclusive, 161, 165, 167, 168, 177, 178, 179, 200 to 202, inclusive, and 205.

The list `PicUnitRepConSeiList` is set to consist of the `payloadType` values 0, 1, 2, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 129, 131, 132, 133, 135 to 152, inclusive, 154 to 168, inclusive, 176 to 180, inclusive, 200 to 202, inclusive, and 205.

The semantics and persistence scope for each SEI message specified in this annex are specified in the semantics specification for each particular SEI message in the subclauses of this clause.

NOTE – Persistence information for SEI messages specified in this annex is informatively summarized in Table G.1.

Table G.1 – Persistence scope of SEI messages (informative)

SEI message	Persistence scope
3D reference displays information	Specified by the semantics of the SEI message
Depth representation information	Specified by the semantics of the SEI message.
Multiview scene information	The CVS containing the SEI message
Multiview acquisition information	The CVS containing the SEI message
Multiview view position SEI message	The CVS containing the SEI message

G.14.3.2 Annex D and Annex F SEI message semantics for multiview high efficiency video coding

G.14.3.2.1 General

The specifications of clause F.14.3.2 and its subclauses apply with the modifications specified in clause G.14.3.2.2.

G.14.3.2.2 Scalable nesting SEI message semantics for multiview high efficiency video coding

The specifications of clause F.14.3.2.7 apply with the following additions:

An SEI message that has `payloadType` equal to 176 (3D reference displays information) or 180 (multiview view position) shall not be directly contained in a scalable nesting SEI message.

When the scalable nesting SEI message contains an SEI message that has `payloadType` equal to 177, 178 or 179, `bitstream_subset_flag` shall be equal to 0.

G.14.3.2.3 3D reference displays information SEI message semantics

A 3D reference displays information SEI message contains information about the reference display width(s) and reference viewing distance(s) as well as information about the corresponding reference stereo pair(s), i.e., the pair(s) of views to be displayed for the viewer's left and right eyes on the reference display at the reference viewing distance. This information enables a view renderer to generate a proper stereo pair for the target screen width and the viewing distance. The reference display width and viewing distance values are signalled in units of centimetres. The reference pair of view specified in this SEI message can be used to extract or infer parameters related to the distance between the camera centres in the reference stereo pair, which can be used for generation of views for the target display. For multi-view displays, the reference stereo pair corresponds to a pair of views that can be simultaneously observed by the viewer's left and right eyes.

When present, this SEI message shall be associated with an IRAP access unit or with a non-IRAP access unit, when all access units that follow this access unit in the decoding order also follow it in output order. The 3D reference display

information SEI message applies to the current access unit and all the access units which follow this access unit in both the output and decoding order until but not including the next IRAP access unit or the next access unit containing a 3D reference displays information SEI message.

NOTE 1 – The 3D reference displays information SEI message specifies display parameters for which the 3D sequence was optimized and the corresponding reference parameters. Each reference display (i.e., a reference display width and possibly a corresponding viewing distance) is associated with one reference pair of views by signalling their ViewId. The difference between the values of ViewId is referred to as the baseline distance (i.e., the distance between the centres of the cameras used to obtain the video sequence).

The following equations can be used for determining the baseline distance and horizontal shift for the receiver's display when the ratio between the receiver's viewing distance and the reference viewing distance is the same as the ratio between the receiver screen width and the reference screen width:

$$\text{baseline}[i] = \text{refBaseline}[i] * (\text{refDisplayWidth}[i] \div \text{displayWidth}) \quad (\text{G-3})$$

$$\text{shift}[i] = \text{refShift}[i] * (\text{refDisplayWidth}[i] \div \text{displayWidth}) \quad (\text{G-4})$$

where $\text{refBaseline}[i]$ is equal to $\text{right_view_id}[i] - \text{left_view_id}[i]$ signalled in this SEI message. Other parameters related to the view generation may be obtained determined by using a similar equation.

$$\text{parameter}[i] = \text{refParameter}[i] * (\text{refDisplayWidth}[i] \div \text{displayWidth}) \quad (\text{G-5})$$

where $\text{refParameter}[i]$ is a parameter related to view generation that corresponds to the reference pair of views signalled by $\text{left_view_id}[i]$ and $\text{right_view_id}[i]$. In the above equations, the width of the visible part of the display used for showing the video sequence should be understood under "display width". The same equations can also be used for determining the pair of views and horizontal shift or other view synthesis parameters when the viewing distance is not scaled proportionally to the screen width compared to the reference display parameters. In this case, the effect of applying the above equations would be to keep the perceived depth in the same proportion to the viewing distance as in the reference setup.

When the view synthesis related parameters that correspond to the reference stereo pair change from one access unit to another, they should be scaled with the same scaling factor as the parameters in the access unit that the SEI message is associated with. Therefore, the above equation should also be applied to obtain the parameters for a following access unit, where the refParameter is the parameter related to the reference stereo pair associated the following access unit.

The horizontal shift for the receiver's display should also be modified by scaling it with the same factor as that used to scale the baseline distance (or other view synthesis parameters).

prec_ref_display_width specifies the exponent of the maximum allowable truncation error for $\text{refDisplayWidth}[i]$ as given by $2^{-\text{prec_ref_display_width}}$. The value of **prec_ref_display_width** shall be in the range of 0 to 31, inclusive.

ref_viewing_distance_flag equal to 1 indicates the presence of reference viewing distance. **ref_viewing_distance_flag** equal to 0 indicates that the reference viewing distance is not present.

prec_ref_viewing_dist specifies the exponent of the maximum allowable truncation error for $\text{refViewingDist}[i]$ as given by $2^{-\text{prec_ref_viewing_dist}}$. The value of **prec_ref_viewing_dist** shall be in the range of 0 to 31, inclusive.

num_ref_displays_minus1 plus 1 specifies the number of reference displays that are signalled in this SEI message. The value of **num_ref_displays_minus1** shall be in the range of 0 to 31, inclusive.

left_view_id[i] indicates the ViewId of the left view of a stereo pair corresponding to the i-th reference display.

right_view_id[i] indicates the ViewId of the right view of a stereo-pair corresponding to the i-th reference display.

exponent_ref_display_width[i] specifies the exponent part of the reference display width of the i-th reference display. The value of **exponent_ref_display_width**[i] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified reference display width.

mantissa_ref_display_width[i] specifies the mantissa part of the reference display width of the i-th reference display. The variable **refDispWidthBits** specifying the number of bits of the **mantissa_ref_display_width**[i] syntax element is derived as follows:

- If **exponent_ref_display_width**[i] is equal to 0, **refDispWidthBits** is set equal to $\text{Max}(0, \text{prec_ref_display_width} - 30)$.
- Otherwise ($0 < \text{exponent_ref_display_width}[i] < 63$), **refDispWidthBits** is set equal to $\text{Max}(0, \text{exponent_ref_display_width}[i] + \text{prec_ref_display_width} - 31)$.

exponent_ref_viewing_distance[i] specifies the exponent part of the reference viewing distance of the i-th reference display. The value of **exponent_ref_viewing_distance**[i] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified reference display width.

mantissa_ref_viewing_distance[i] specifies the mantissa part of the reference viewing distance of the i-th reference display. The variable **refViewDistBits** specifying the number of bits of the **mantissa_ref_viewing_distance**[i] syntax element is derived as follows:

- If `exponent_ref_viewing_distance[i]` is equal to 0, the `refViewDistBits` is set equal to `Max(0, prec_ref_viewing_distance – 30)`.
- Otherwise (`0 < exponent_ref_viewing_distance[i] < 63`), `refViewDistBits` is set equal to `Max(0, exponent_ref_viewing_distance[i] + prec_ref_viewing_distance – 31)`.

The variables in the x row of Table G.2 are derived from the respective variables or values in the e, n and v rows of Table G.2 as follows:

- If e is not equal to 0, the following applies:

$$x = 2^{(e-31)} * (1 + n \div 2^v) \quad (G-6)$$

- Otherwise (e is equal to 0), the following applies:

$$x = 2^{-(30+v)} * n \quad (G-7)$$

NOTE 2 – The above specification is similar to that found in IEC 60559:1989.

Table G.2 – Association between camera parameter variables and syntax elements

x	<code>refDisplayWidth[i]</code>	<code>refViewingDistance[i]</code>
e	<code>exponent_ref_display_width[i]</code>	<code>exponent_ref_viewing_distance[i]</code>
n	<code>mantissa_ref_display_width[i]</code>	<code>mantissa_ref_viewing_distance[i]</code>
v	<code>refDispWidthBits</code>	<code>refViewDistBits</code>

additional_shift_present_flag[i] equal to 1 indicates that the information about additional horizontal shift of the left and right views for the i-th reference display is present in this SEI message. **additional_shift_present_flag[i]** equal to 0 indicates that the information about additional horizontal shift of the left and right views for the i-th reference display is not present in this SEI message.

num_sample_shift_plus512[i] indicates the recommended additional horizontal shift for a stereo pair corresponding to the i-th reference baseline and the i-th reference display.

- If `num_sample_shift_plus512[i]` is less than 512, it is recommended that the left view of the stereo pair corresponding to the i-th reference baseline and the i-th reference display is shifted in the left direction by (`512 – num_sample_shift_plus512[i]`) samples with respect to the right view of the stereo pair.
- Otherwise, if `num_sample_shift_plus512[i]` is equal to 512, it is recommended that shifting is not applied.
- Otherwise, (`num_sample_shift_plus512[i]` is greater than 512), it is recommended that the left view in the stereo pair corresponding to the i-th reference baseline and the i-th reference display should be shifted in the right direction by (`num_sample_shift_plus512[i] – 512`) samples with respect to the right view of the stereo pair.

The value of `num_sample_shift_plus512[i]` shall be in the range of 0 to 1 023, inclusive.

NOTE 3 – Shifting the left view in the left (or right) direction by x samples with respect to the right view can be performed by the following two-step processing:

- 1) Shift the left view by `x / 2` samples in the left (or right) direction and shift the right view by `x / 2` samples in the right (or left) direction.
- 2) Fill the left and right image margins of `x / 2` samples in width in both the left and right views in background colour.

The following explains the recommended shifting processing in the case of shifting the left view in the left direction by x samples with respect to the right view:

```

for( i = x / 2; i < width - x / 2; i++ )
    for( j = 0; j < height; j++ ) {
        leftView[ j ][ i ] = leftView[ j ][ i + x / 2 ]
        rightView[ j ][ width - 1 - i ] = rightView[ j ][ width - 1 - i - x / 2 ]
    }
for( i = 0; i < x / 2; i++ )
    for( j = 0; j < height; j++ ) {
        leftView[ j ][ width - 1 - i ] = leftView[ j ][ i ] = backgroundColour
        rightView[ j ][ width - 1 - i ] = rightView[ j ][ i ] = backgroundColour
    }

```

(G-8)

The following explains the recommended shifting processing in the case of shifting the left view in the right direction by x samples with respect to the right view:

```

for( i = x / 2; i < width - x / 2; i++ )
    for( j = 0; j < height; j++ ) {
        leftView[ j ][ width - 1 - i ] = leftView[ j ][ width - 1 - i - x / 2 ]
        rightView[ j ][ i ] = rightView[ j ][ i + x / 2 ]
    }
for( i = 0; i < x / 2; i++ )
    for( j = 0; j < height; j++ ) {
        leftView[ j ][ width - 1 - i ] = leftView[ j ][ i ] = backgroundColour
        rightView[ j ][ width - 1 - i ] = rightView[ j ][ i ] = backgroundColour
    }

```

(G-9)

The variable backgroundColour may take different values in different systems, for example black or grey.

three_dimensional_reference_displays_extension_flag equal to 0 indicates that no additional data follows within the reference displays SEI message. The value of three_dimensional_reference_displays_extension_flag shall be equal to 0 in bitstreams conforming to this version of this Specification. The value of 1 for three_dimensional_reference_displays_extension_flag is reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore all data that follow the value 1 for three_dimensional_reference_displays_extension_flag in a reference displays SEI message.

G.14.3.3 Depth representation information SEI message semantics

G.14.3.3.1 General

The syntax elements in the depth representation information SEI message specify various parameters for auxiliary pictures of type AUX_DEPTH for the purpose of processing decoded primary and auxiliary pictures prior to rendering on a 3D display, such as view synthesis. Specifically, depth or disparity ranges for depth pictures are specified.

When present, the depth representation information SEI message shall be associated with one or more layers with AuxId value equal to AUX_DEPTH. The following semantics apply separately to each nuh_layer_id targetLayerId among the nuh_layer_id values to which the depth representation information SEI message applies.

When present, the depth representation information SEI message may be included in any access unit. It is recommended that, when present, the SEI message is included for the purpose of random access in an access unit in which the coded picture with nuh_layer_id equal to targetLayerId is an IRAP picture.

For an auxiliary picture with AuxId[targetLayerId] equal to AUX_DEPTH, an associated primary picture, if any, is a picture in the same access unit having AuxId[nuhLayerIdB] equal to 0 such that ScalabilityId[LayerIdxInVps[targetLayerId]][j] is equal to ScalabilityId[LayerIdxInVps[nuhLayerIdB]][j] for all values of j in the range of 0 to 2, inclusive, and 4 to 15, inclusive.

The information indicated in the SEI message applies to all the pictures with nuh_layer_id equal to targetLayerId from the access unit containing the SEI message up to but excluding the next picture, in decoding order, associated with a depth representation information SEI message applicable to targetLayerId or to the end of the CLVS of the nuh_layer_id equal to targetLayerId, whichever is earlier in decoding order.

z_near_flag equal to 0 specifies that the syntax elements specifying the nearest depth value are not present in the syntax structure. z_near_flag equal to 1 specifies that the syntax elements specifying the nearest depth value are present in the syntax structure.

z_far_flag equal to 0 specifies that the syntax elements specifying the farthest depth value are not present in the syntax structure. z_far_flag equal to 1 specifies that the syntax elements specifying the farthest depth value are present in the syntax structure.

d_min_flag equal to 0 specifies that the syntax elements specifying the minimum disparity value are not present in the syntax structure. d_min_flag equal to 1 specifies that the syntax elements specifying the minimum disparity value are present in the syntax structure.

d_max_flag equal to 0 specifies that the syntax elements specifying the maximum disparity value are not present in the syntax structure. d_max_flag equal to 1 specifies that the syntax elements specifying the maximum disparity value are present in the syntax structure.

depth_representation_type specifies the representation definition of decoded luma samples of auxiliary pictures as specified in Table G.3. In Table G.3, disparity specifies the horizontal displacement between two texture views and Z value specifies the distance from a camera.

The variable maxVal is set equal to $(1 \ll (8 + \text{bit_depth_luma_minus8})) - 1$, where bit_depth_luma_minus8 is the value included in or inferred for the active SPS of the layer with nuh_layer_id equal to targetLayerId. The value of

depth_representation_type shall be in the range of 0 to 3, inclusive, in bitstreams conforming to this version of this Specification. The values of 4 to 15, inclusive, for depth_representation_type are reserved for future use by ITU-T | ISO/IEC. Although the value of depth_representation_type is required to be in the range of 0 to 3, inclusive, in this version of this Specification, decoders shall allow values of depth_representation_type in the range of 4 to 15, inclusive, to appear in the syntax. Decoders conforming to this version of this Specification shall ignore all data that follow a value of depth_representation_type in the range of 4 to 15, inclusive, in the depth representation information SEI message.

Table G.3 – Definition of depth_representation_type

depth_representation_type	Interpretation
0	Each decoded luma sample value of an auxiliary picture represents an inverse of Z value that is uniformly quantized into the range of 0 to maxVal, inclusive. When z_far_flag is equal to 1, the luma sample value equal to 0 represents the inverse of ZFar (specified below). When z_near_flag is equal to 1, the luma sample value equal to maxVal represents the inverse of ZNear (specified below).
1	Each decoded luma sample value of an auxiliary picture represents disparity that is uniformly quantized into the range of 0 to maxVal, inclusive. When d_min_flag is equal to 1, the luma sample value equal to 0 represents DMin (specified below). When d_max_flag is equal to 1, the luma sample value equal to maxVal represents DMax (specified below).
2	Each decoded luma sample value of an auxiliary picture represents a Z value uniformly quantized into the range of 0 to maxVal, inclusive. When z_far_flag is equal to 1, the luma sample value equal to 0 corresponds to ZFar (specified below). When z_near_flag is equal to 1, the luma sample value equal to maxVal represents ZNear (specified below).
3	Each decoded luma sample value of an auxiliary picture represents a non-linearly mapped disparity, normalized in range from 0 to maxVal, as specified by depth_nonlinear_representation_num_minus1 and depth_nonlinear_representation_model[i]. When d_min_flag is equal to 1, the luma sample value equal to 0 represents DMin (specified below). When d_max_flag is equal to 1, the luma sample value equal to maxVal represents DMax (specified below).
4..15	Reserved

disparity_ref_view_id specifies the ViewId value for which the disparity values are derived. The value of disparity_ref_view_id shall be in the range of 0 to 1 023, inclusive.

NOTE 1 – disparity_ref_view_id is present only if d_min_flag is equal to 1 or d_max_flag is equal to 1 and is useful for depth_representation_type values equal to 1 and 3.

The variables in the x column of Table G.4 are derived from the respective variables in the s, e, n and v columns of Table G.4 as follows:

- If the value of e is in the range of 0 to 127, exclusive, x is set equal to $(-1)^s * 2^{e-31} * (1 + n \div 2^v)$.
- Otherwise (e is equal to 0), x is set equal to $(-1)^s * 2^{-(30+v)} * n$.

NOTE 1 – The above specification is similar to that found in IEC 60559:1989.

Table G.4 – Association between depth parameter variables and syntax elements

x	S	e	n	v
ZNear	ZNearSign	ZNearExp	ZNearMantissa	ZNearManLen
ZFar	ZFarSign	ZFarExp	ZFarMantissa	ZFarManLen
DMax	DMaxSign	DMaxExp	DMaxMantissa	DMaxManLen
DMin	DMinSign	DMinExp	DMinMantissa	DMinManLen

The DMin and DMax values, when present, are specified in units of a luma sample width of the coded picture with ViewId equal to ViewId of the auxiliary picture.

The units for the ZNear and ZFar values, when present, are identical but unspecified.

depth_nonlinear_representation_num_minus1 plus 2 specifies the number of piece-wise linear segments for mapping of depth values to a scale that is uniformly quantized in terms of disparity. The value of **depth_nonlinear_representation_num_minus1** shall be in the range of 0 to 62, inclusive.

depth_nonlinear_representation_model[i] for *i* ranging from 0 to **depth_nonlinear_representation_num_minus1 + 2**, inclusive, specify the piece-wise linear segments for mapping of decoded luma sample values of an auxiliary picture to a scale that is uniformly quantized in terms of disparity. The value of **depth_nonlinear_representation_model[i]** shall be in the range of 0 to 65 535, inclusive. The values of **depth_nonlinear_representation_model[0]** and **depth_nonlinear_representation_model[depth_nonlinear_representation_num_minus1 + 2]** are both inferred to be equal to 0.

NOTE 2 – When **depth_representation_type** is equal to 3, an auxiliary picture contains non-linearly transformed depth samples. The variable **DepthLUT[i]**, as specified below, is used to transform decoded depth sample values from the non-linear representation to the linear representation, i.e., uniformly quantized disparity values. The shape of this transform is defined by means of line-segment approximation in two-dimensional linear-disparity-to-non-linear-disparity space. The first (0, 0) and the last (**maxVal**, **maxVal**) nodes of the curve are predefined. Positions of additional nodes are transmitted in form of deviations (**depth_nonlinear_representation_model[i]**) from the straight-line curve. These deviations are uniformly distributed along the whole range of 0 to **maxVal**, inclusive, with spacing depending on the value of **nonlinear_depth_representation_num_minus1**.

The variable **DepthLUT[i]** for *i* in the range of 0 to **maxVal**, inclusive, is specified as follows:

```
for( k = 0; k <= depth_nonlinear_representation_num_minus1 + 1; k++ ) {
    pos1 = ( maxVal * k ) / (depth_nonlinear_representation_num_minus1 + 2 )
    dev1 = depth_nonlinear_representation_model[ k ]
    pos2 = ( maxVal * ( k + 1 ) ) / (depth_nonlinear_representation_num_minus1 + 2 )
    dev2 = depth_nonlinear_representation_model[ k + 1 ]
    x1 = pos1 - dev1
    y1 = pos1 + dev1
    x2 = pos2 - dev2
    y2 = pos2 + dev2

    for( x = Max( x1, 0 ); x <= Min( x2, maxVal ); x++ )
        DepthLUT[ x ] = Clip3( 0, maxVal, Round( (( x - x1 ) * ( y2 - y1 )) ÷ ( x2 - x1 ) + y1 ) )
}
```

(G-10)

When **depth_representation_type** is equal to 3, **DepthLUT[dS]** for all decoded luma sample values **dS** of an auxiliary picture in the range of 0 to **maxVal**, inclusive, represents disparity that is uniformly quantized into the range of 0 to **maxVal**, inclusive.

G.14.3.3.2 Depth representation information element semantics

The syntax structure specifies the value of an element in the depth representation information SEI message.

The **depth_rep_info_element(OutSign, OutExp, OutMantissa, OutManLen)** syntax structure sets the values of the **OutSign**, **OutExp**, **OutMantissa** and **OutManLen** variables that represent a floating-point value. When the syntax structure is included in another syntax structure, the variable names **OutSign**, **OutExp**, **OutMantissa** and **OutManLen** are to be interpreted as being replaced by the variable names used when the syntax structure is included.

da_sign_flag equal to 0 indicates that the sign of the floating-point value is positive. **da_sign_flag** equal to 1 indicates that the sign is negative. The variable **OutSign** is set equal to **da_sign_flag**.

da_exponent specifies the exponent of the floating-point value. The value of **da_exponent** shall be in the range of 0 to $2^7 - 2$, inclusive. The value $2^7 - 1$ is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value $2^7 - 1$ as indicating an unspecified value. The variable **OutExp** is set equal to **da_exponent**.

da_mantissa_len_minus1 plus 1 specifies the number of bits in the **da_mantissa** syntax element. The variable **OutManLen** is set equal to **da_mantissa_len_minus1 + 1**.

da_mantissa specifies the mantissa of the floating-point value. The variable **OutMantissa** is set equal to **da_mantissa**.

G.14.3.4 Multiview scene information SEI message semantics

The multiview scene information SEI message indicates the minimum disparity and the maximum disparity among multiple views in an access unit. The minimum disparity and the maximum disparity could be used for processing the decoded views prior to rendering on a 3D display. When present, the multiview scene information SEI message shall be associated with an IRAP access unit. The information signalled in the SEI message applies to the coded video sequence. If the SEI message is not contained within a scalable nesting SEI message, it applies to the views with **nuh_layer_id** greater than or equal to the **nuh_layer_id** value of the SEI NAL unit that contains the SEI message. Otherwise (the SEI message is contained within a scalable nesting SEI message), the SEI message applies to the view with **nuh_layer_id** values identified in the scalable nesting SEI message. The views to which the SEI message applies are referred to as applicable views below.

The actual minimum disparity value may be greater than the one signalled in the multiview scene information SEI message and the actual maximum disparity value may be less than the one signalled in the multiview scene information SEI message, due to that some views in the coded video sequence may have been removed from the original bitstream to produce an extracted sub-bitstream, for example according to the process specified in clause 10.

Let x_R be a luma sample position within a luma sample array of the right-side picture of any spatially adjacent views among the applicable views in an access unit. Let x_L be the respective luma sample position within a luma sample array of the left-side picture of the same spatially adjacent view such that sample in the luma sample position x_L in the left-side view represents the same content as luma sample position x_R in the right-side picture. When $\text{pic_width_in_luma_samples}$ for the left-side picture, picWidthL , is not equal to $\text{pic_width_in_luma_samples}$ for the right-side picture, picWidthR , x_L is normalized to the horizontal luma sample resolution of the right-side picture, i.e., x_L is set equal to $\text{Round}(x_L * (\text{picWidthR} \div \text{picWidthL}))$. The disparity between x_R and x_L is specified to be equal to $(x_R - x_L)$. The maximum disparity between two pictures is defined to be the maximum of the disparity between any sample position pairs x_R and x_L . The minimum disparity between two pictures is defined to be the minimum of the disparity between any sample position pairs x_R and x_L .

min_disparity specifies the minimum disparity, in units of luma samples, between pictures of any spatially adjacent views among the applicable views in an access unit. The value of **min_disparity** shall be in the range of $-1\,024$ to $1\,023$, inclusive.

max_disparity_range specifies that the maximum disparity, in units of luma samples, between pictures of any spatially adjacent views among the applicable views in an access unit. The value of **max_disparity_range** shall be in the range of 0 to 2047, inclusive.

NOTE – The minimum disparity and the maximum disparity depend on the baseline distance between spatially adjacent views and the spatial resolution of each view. Therefore, if either the number of views or spatial resolution is changed, the minimum disparity and the maximum disparity should also be changed accordingly.

G.14.3.5 Multiview acquisition information SEI message semantics

The multiview acquisition information SEI message specifies various parameters of the acquisition environment. Specifically, intrinsic and extrinsic camera parameters are specified. These parameters could be used for processing the decoded views prior to rendering on a 3D display.

The following semantics apply separately to each nuh_layer_id `targetLayerId` among the nuh_layer_id values to which the multiview acquisition information SEI message applies as specified in clause D.3.1.

When present, the multiview acquisition information SEI message that applies to the current layer shall be included in an access unit that contains an IRAP picture that is the first picture of a CLVS of the current layer. The information signalled in the SEI message applies to the CLVS.

When the multiview acquisition information SEI message is included in a scalable nesting SEI message, the syntax elements `bitstream_subset_flag`, `nesting_op_flag` and `all_layers_flag` in the scalable nesting SEI message shall be equal to 0.

The variable `numViewsMinus1` is derived as follows:

- If the multiview acquisition information SEI message is not included in a scalable nesting SEI message, `numViewsMinus1` is set equal to 0.
- Otherwise (the multiview acquisition information SEI message is included in a scalable nesting SEI message), `numViewsMinus1` is set equal to `nesting_num_layers_minus1`.

Some of the views for which the multiview acquisition information is included in a multiview acquisition information SEI message may not be present.

In the semantics below, index i refers to the syntax elements and variables that apply to the layer with nuh_layer_id equal to `nestingLayerIdList[0][i]`.

The extrinsic camera parameters are specified according to a right-handed coordinate system, where the upper left corner of the image is the origin, i.e., the $(0, 0)$ coordinate, with the other corners of the image having non-negative coordinates. With these specifications, a 3-dimensional world point, $wP = [x\ y\ z]$ is mapped to a 2-dimensional camera point, $cP[i] = [u\ v\ 1]$, for the i -th camera according to:

$$s * cP[i] = A[i] * R^{-1}[i] * (wP - T[i]) \quad (G-11)$$

where $A[i]$ denotes the intrinsic camera parameter matrix, $R^{-1}[i]$ denotes the inverse of the rotation matrix $R[i]$, $T[i]$ denotes the translation vector and s (a scalar value) is an arbitrary scale factor chosen to make the third coordinate of $cP[i]$ equal to 1. The elements of $A[i]$, $R[i]$ and $T[i]$ are determined according to the syntax elements signalled in this SEI message and as specified below.

intrinsic_param_flag equal to 1 indicates the presence of intrinsic camera parameters. **intrinsic_param_flag** equal to 0

indicates the absence of intrinsic camera parameters.

extrinsic_param_flag equal to 1 indicates the presence of extrinsic camera parameters. **extrinsic_param_flag** equal to 0 indicates the absence of extrinsic camera parameters.

intrinsic_params_equal_flag equal to 1 indicates that the intrinsic camera parameters are equal for all cameras and only one set of intrinsic camera parameters is present. **intrinsic_params_equal_flag** equal to 0 indicates that the intrinsic camera parameters are different for each camera and that a set of intrinsic camera parameters is present for each camera.

prec_focal_length specifies the exponent of the maximum allowable truncation error for **focal_length_x[i]** and **focal_length_y[i]** as given by $2^{-\text{prec_focal_length}}$. The value of **prec_focal_length** shall be in the range of 0 to 31, inclusive.

prec_principal_point specifies the exponent of the maximum allowable truncation error for **principal_point_x[i]** and **principal_point_y[i]** as given by $2^{-\text{prec_principal_point}}$. The value of **prec_principal_point** shall be in the range of 0 to 31, inclusive.

prec_skew_factor specifies the exponent of the maximum allowable truncation error for skew factor as given by $2^{-\text{prec_skew_factor}}$. The value of **prec_skew_factor** shall be in the range of 0 to 31, inclusive.

sign_focal_length_x[i] equal to 0 indicates that the sign of the focal length of the i-th camera in the horizontal direction is positive. **sign_focal_length_x[i]** equal to 1 indicates that the sign is negative.

exponent_focal_length_x[i] specifies the exponent part of the focal length of the i-th camera in the horizontal direction. The value of **exponent_focal_length_x[i]** shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified focal length.

mantissa_focal_length_x[i] specifies the mantissa part of the focal length of the i-th camera in the horizontal direction. The length of the **mantissa_focal_length_x[i]** syntax element in units of bits is variable and determined as follows:

- If **exponent_focal_length_x[i]** is equal to 0, the length is $\text{Max}(0, \text{prec_focal_length} - 30)$.
- Otherwise (**exponent_focal_length_x[i]** is in the range of 0 to 63, exclusive), the length is $\text{Max}(0, \text{exponent_focal_length_x}[i] + \text{prec_focal_length} - 31)$.

sign_focal_length_y[i] equal to 0 indicates that the sign of the focal length of the i-th camera in the vertical direction is positive. **sign_focal_length_y[i]** equal to 1 indicates that the sign is negative.

exponent_focal_length_y[i] specifies the exponent part of the focal length of the i-th camera in the vertical direction. The value of **exponent_focal_length_y[i]** shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified focal length.

mantissa_focal_length_y[i] specifies the mantissa part of the focal length of the i-th camera in the vertical direction.

The length of the **mantissa_focal_length_y[i]** syntax element in units of bits is variable and determined as follows:

- If **exponent_focal_length_y[i]** is equal to 0, the length is $\text{Max}(0, \text{prec_focal_length} - 30)$.
- Otherwise (**exponent_focal_length_y[i]** is in the range of 0 to 63, exclusive), the length is $\text{Max}(0, \text{exponent_focal_length_y}[i] + \text{prec_focal_length} - 31)$.

sign_principal_point_x[i] equal to 0 indicates that the sign of the principal point of the i-th camera in the horizontal direction is positive. **sign_principal_point_x[i]** equal to 1 indicates that the sign is negative.

exponent_principal_point_x[i] specifies the exponent part of the principal point of the i-th camera in the horizontal direction. The value of **exponent_principal_point_x[i]** shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified principal point.

mantissa_principal_point_x[i] specifies the mantissa part of the principal point of the i-th camera in the horizontal direction. The length of the **mantissa_principal_point_x[i]** syntax element in units of bits is variable and is determined as follows:

- If **exponent_principal_point_x[i]** is equal to 0, the length is $\text{Max}(0, \text{prec_principal_point} - 30)$.
- Otherwise (**exponent_principal_point_x[i]** is in the range of 0 to 63, exclusive), the length is $\text{Max}(0, \text{exponent_principal_point_x}[i] + \text{prec_principal_point} - 31)$.

sign_principal_point_y[i] equal to 0 indicates that the sign of the principal point of the i-th camera in the vertical direction is positive. **sign_principal_point_y[i]** equal to 1 indicates that the sign is negative.

exponent_principal_point_y[i] specifies the exponent part of the principal point of the i-th camera in the vertical direction. The value of **exponent_principal_point_y[i]** shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified principal point.

mantissa_principal_point_y[i] specifies the mantissa part of the principal point of the i-th camera in the vertical direction. The length of the mantissa_principal_point_y[i] syntax element in units of bits is variable and is determined as follows:

- If exponent_principal_point_y[i] is equal to 0, the length is $\text{Max}(0, \text{prec_principal_point} - 30)$.
- Otherwise (exponent_principal_point_y[i] is in the range of 0 to 63, exclusive), the length is $\text{Max}(0, \text{exponent_principal_point_y}[i] + \text{prec_principal_point} - 31)$.

sign_skew_factor[i] equal to 0 indicates that the sign of the skew factor of the i-th camera is positive.

sign_skew_factor[i] equal to 1 indicates that the sign is negative.

exponent_skew_factor[i] specifies the exponent part of the skew factor of the i-th camera. The value of exponent_skew_factor[i] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified skew factor.

mantissa_skew_factor[i] specifies the mantissa part of the skew factor of the i-th camera. The length of the mantissa_skew_factor[i] syntax element in units of bits is variable and determined as follows:

- If exponent_skew_factor[i] is equal to 0, the length is $\text{Max}(0, \text{prec_skew_factor} - 30)$.
- Otherwise (exponent_skew_factor[i] is in the range of 0 to 63, exclusive), the length is $\text{Max}(0, \text{exponent_skew_factor}[i] + \text{prec_skew_factor} - 31)$.

The intrinsic matrix $A[i]$ for i-th camera is represented by

$$\begin{bmatrix} \text{focalLengthX}[i] & \text{skewFactor}[i] & \text{principalPointX}[i] \\ 0 & \text{focalLengthY}[i] & \text{principalPointY}[i] \\ 0 & 0 & 1 \end{bmatrix} \quad (\text{G-12})$$

prec_rotation_param specifies the exponent of the maximum allowable truncation error for $r[i][j][k]$ as given by $2^{-\text{prec_rotation_param}}$. The value of prec_rotation_param shall be in the range of 0 to 31, inclusive.

prec_translation_param specifies the exponent of the maximum allowable truncation error for $t[i][j]$ as given by $2^{-\text{prec_translation_param}}$. The value of prec_translation_param shall be in the range of 0 to 31, inclusive.

sign_r[i][j][k] equal to 0 indicates that the sign of (j, k) component of the rotation matrix for the i-th camera is positive. **sign_r[i][j][k]** equal to 1 indicates that the sign is negative.

exponent_r[i][j][k] specifies the exponent part of (j, k) component of the rotation matrix for the i-th camera. The value of exponent_r[i][j][k] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified rotation matrix.

mantissa_r[i][j][k] specifies the mantissa part of (j, k) component of the rotation matrix for the i-th camera. The length of the mantissa_r[i][j][k] syntax element in units of bits is variable and determined as follows:

- If exponent_r[i] is equal to 0, the length is $\text{Max}(0, \text{prec_rotation_param} - 30)$.
- Otherwise (exponent_r[i] is in the range of 0 to 63, exclusive), the length is $\text{Max}(0, \text{exponent_r}[i] + \text{prec_rotation_param} - 31)$.

The rotation matrix $R[i]$ for i-th camera is represented as follows:

$$\begin{bmatrix} rE[i][0][0] & rE[i][0][1] & rE[i][0][2] \\ rE[i][1][0] & rE[i][1][1] & rE[i][1][2] \\ rE[i][2][0] & rE[i][2][1] & rE[i][2][2] \end{bmatrix} \quad (\text{G-13})$$

sign_t[i][j] equal to 0 indicates that the sign of the j-th component of the translation vector for the i-th camera is positive. **sign_t[i][j]** equal to 1 indicates that the sign is negative.

exponent_t[i][j] specifies the exponent part of the j-th component of the translation vector for the i-th camera. The value of exponent_t[i][j] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. Decoders shall treat the value 63 as indicating an unspecified translation vector.

mantissa_t[i][j] specifies the mantissa part of the j-th component of the translation vector for the i-th camera. The length v of the mantissa_t[i][j] syntax element in units of bits is variable and is determined as follows:

- If $\text{exponent_t}[i]$ is equal to 0, the length v is set equal to $\text{Max}(0, \text{prec_translation_param} - 30)$.
- Otherwise ($0 < \text{exponent_t}[i] < 63$), the length v is set equal to $\text{Max}(0, \text{exponent_t}[i] + \text{prec_translation_param} - 31)$.

The translation vector $T[i]$ for the i -th camera is represented by:

$$\begin{bmatrix} tE[i][0] \\ tE[i][1] \\ tE[i][2] \end{bmatrix} \quad (\text{G-14})$$

The association between the camera parameter variables and corresponding syntax elements is specified by Table G.5. Each component of the intrinsic and rotation matrices and the translation vector is obtained from the variables specified in Table G.5 as the variable x computed as follows:

- If e is in the range of 0 to 63, exclusive, x is set equal to $(-1)^s * 2^{e-31} * (1 + n \div 2^v)$.
- Otherwise (e is equal to 0), x is set equal to $(-1)^s * 2^{-(30+v)} * n$.

NOTE – The above specification is similar to that found in IEC 60559:1989.

Table G.5 – Association between camera parameter variables and syntax elements.

x	s	e	n
focalLengthX [i]	sign_focal_length_x[i]	exponent_focal_length_x[i]	mantissa_focal_length_x[i]
focalLengthY [i]	sign_focal_length_y[i]	exponent_focal_length_y[i]	mantissa_focal_length_y[i]
principalPointX [i]	sign_principal_point_x[i]	exponent_principal_point_x[i]	mantissa_principal_point_x[i]
principalPointY [i]	sign_principal_point_y[i]	exponent_principal_point_y[i]	mantissa_principal_point_y[i]
skewFactor [i]	sign_skew_factor[i]	exponent_skew_factor[i]	mantissa_skew_factor[i]
rE [i][j][k]	sign_r[i][j][k]	exponent_r[i][j][k]	mantissa_r[i][j][k]
tE [i][j]	sign_t[i][j]	exponent_t[i][j]	mantissa_t[i][j]

G.14.3.6 Multiview view position SEI message semantics

The multiview view position SEI message specifies the relative view position along a single horizontal axis of views within a CVS. When present, the multiview view position SEI message shall be associated with an IRAP access unit. The information signalled in this SEI message applies to the entire CVS.

num_views_minus1 plus 1 shall be equal to NumViews derived from the active VPS for the CVS. The value of **num_views_minus1** shall be in the range of 0 to 62, inclusive.

view_position[i] indicates the order of the view with ViewOrderIdx equal to i among all the views from left to right for the purpose of display, with the order for the left-most view being equal to 0 and the value of the order increasing by 1 for next view from left to right. The value of **view_position**[i] shall be in the range of 0 to 62, inclusive.

G.15 Video usability information

The specifications in clause F.15 and its subclauses apply.

Annex H

Scalable high efficiency video coding

(This annex forms an integral part of this Recommendation | International Standard.)

H.1 Scope

This annex specifies syntax, semantics and decoding processes for scalable high efficiency video coding that use the syntax, semantics, and decoding process specified in clauses 2-10 and Annexes A-F. This annex also specifies profiles, tier and levels for scalable high efficiency video coding.

H.2 Normative references

The list of normative references in clause F.2 applies.

H.3 Definitions

The specifications in clause F.3 and its subclauses apply.

H.4 Abbreviations

The specifications in clause F.4 apply.

H.5 Conventions

The specifications in clause F.5 apply.

H.6 Bitstream and picture formats, partitionings, scanning processes, and neighbouring relationships

The specifications in clause F.6 apply.

H.7 Syntax and semantics

The specifications in clause F.7 and its subclauses apply.

H.8 Decoding processes

H.8.1 General decoding process

H.8.1.1 General

The specifications of clause F.8.1.1 apply.

H.8.1.2 Decoding process for a coded picture with `nuh_layer_id` greater than 0

The decoding process for the current picture CurrPic is as follows:

1. The decoding of NAL units is specified in clause H.8.2.
2. The processes in clauses H.8.1.3 and H.8.3.4 specify the following decoding processes using syntax elements in the slice segment layer and above:
 - At the beginning of the decoding process for the first slice of the current picture, the process specified in clause H.8.1.3 is invoked.
 - At the beginning of the decoding process for each P or B slice, the decoding process for reference picture lists construction specified in clause H.8.3.4 is invoked for derivation of reference picture list 0 (RefPicList0), and when decoding a B slice, reference picture list 1 (RefPicList1).
3. The processes in clauses H.8.4, H.8.5, H.8.6 and H.8.7 specify decoding processes using syntax elements in all syntax structure layers. It is a requirement of bitstream conformance that the coded slices of the picture shall contain slice segment data for every CTU of the picture, such that the division of the picture into slices, the division of the slices into slice segments, and the division of the slice segments into CTUs each form a partitioning of the picture.

H.8.1.3 Decoding process for inter-layer reference picture set

Outputs of this process are updated lists of inter-layer reference pictures RefPicSetInterLayer0 and RefPicSetInterLayer1 and the variables NumActiveRefLayerPics0 and NumActiveRefLayerPics1.

The variable currLayerId is set equal to nuh_layer_id of the current picture.

The variables NumRefLayerPicsProcessing, NumRefLayerPicsSampleProcessing and NumRefLayerPicsMotionProcessing are set equal to 0.

The lists RefPicSetInterLayer0 and RefPicSetInterLayer1 are first emptied, NumActiveRefLayerPics0 and NumActiveRefLayerPics1 are set equal to 0 and the following applies:

```
for( i = 0; i < NumActiveRefLayerPics; i++ ) {
    refPicSet0Flag =
        ( ( ViewId[ currLayerId ] <= ViewId[ 0 ] && ViewId[ currLayerId ] <=
ViewId[ RefPicLayerId[ i ] ] ) ||
        ( ViewId[ currLayerId ] >= ViewId[ 0 ] && ViewId[ currLayerId ] >=
ViewId[ RefPicLayerId[ i ] ] ) )
    if( there is a picture picX in the DPB that is in the same access unit as the current picture and has
        nuh_layer_id equal to RefPicLayerId[ i ] ) {
        an inter-layer reference picture ilRefPic is derived by invoking the process specified in clause
H.8.1.4
        with picX and RefPicLayerId[ i ] given as inputs
        if( refPicSet0Flag ) {
            (H-1)
            RefPicSetInterLayer0[ NumActiveRefLayerPics0 ] = ilRefPic
            RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] is marked as "used for long-term
reference"
        } else {
            RefPicSetInterLayer1[ NumActiveRefLayerPics1 ] = ilRefPic
            RefPicSetInterLayer1[ NumActiveRefLayerPics1++ ] is marked as "used for long-term
reference"
        }
    } else {
        if( refPicSet0Flag )
            RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] = "no reference picture"
        else
            RefPicSetInterLayer1[ NumActiveRefLayerPics1++ ] = "no reference picture"
    }
}
```

There shall be no entry equal to "no reference picture" in RefPicSetInterLayer0 or RefPicSetInterLayer1.

There shall be no picture that has discardable_flag equal to 1 in RefPicSetInterLayer0 or RefPicSetInterLayer1.

NOTE 1 – For the profiles defined in this annex, RefPicSetInterLayer1 is always empty since the value of ViewId[i] is equal to zero for all layers.

If the current picture is a RADL picture, there shall be no entry in the RefPicSetInterLayer0 or RefPicSetInterLayer1 that is a RASL picture.

NOTE 2 – An access unit may contain both RASL and RADL pictures.

H.8.1.4 Derivation process for inter-layer reference pictures

H.8.1.4.1 General

Inputs to this process are:

- a decoded direct reference layer picture rLPic,
- a variable rLId specifying the value of nuh_layer_id of the direct reference layer picture.

Output of this process is the inter-layer reference picture ilRefPic.

The variables `PicWidthInSamplesCurrY`, `PicHeightInSamplesCurrY`, `BitDepthCurrY`, `BitDepthCurrC`, `SubWidthCurrC` and `SubHeightCurrC` are set equal to `pic_width_in_luma_samples`, `pic_height_in_luma_samples`, `BitDepthY`, `BitDepthC`, `SubWidthC` and `SubHeightC` of the current layer, respectively.

The variables `PicWidthInSamplesRefLayerY` and `PicHeightInSamplesRefLayerY` are set equal to the width and height of the decoded direct reference layer picture `rlPic` in units of luma samples, respectively. The variables `BitDepthRefLayerY`, `BitDepthRefLayerC`, `SubWidthRefLayerC` and `SubHeightRefLayerC` are set equal to the values of `BitDepthY`, `BitDepthC`, `SubWidthC` and `SubHeightC` of the direct reference layer picture `rlPic`, respectively.

NOTE – This clause and its subclauses support all possible values of `SubWidthCurrC`, `SubHeightCurrC`, `SubWidthRefLayerC` and `SubHeightRefLayerC`. When the value of `chroma_format_idc` of the current layer and the value of `chroma_format_idc` of the direct reference layer are both equal to 1, the values of `SubWidthCurrC`, `SubHeightCurrC`, `SubWidthRefLayerC` and `SubHeightRefLayerC` are all equal to 2, and the resampling process of picture sample values in clause H.8.1.4.2 and the colour mapping process of picture sample values in clause H.8.1.4.4 can be simplified.

The variables `RefLayerRegionLeftOffset`, `RefLayerRegionTopOffset`, `RefLayerRegionRightOffset` and `RefLayerRegionBottomOffset` are derived as follows:

$$\text{RefLayerRegionLeftOffset} = \text{ref_region_left_offset}[\text{rLId}] * \text{SubWidthRefLayerC} \quad (\text{H-2})$$

$$\text{RefLayerRegionTopOffset} = \text{ref_region_top_offset}[\text{rLId}] * \text{SubHeightRefLayerC} \quad (\text{H-3})$$

$$\text{RefLayerRegionRightOffset} = \text{ref_region_right_offset}[\text{rLId}] * \text{SubWidthRefLayerC} \quad (\text{H-4})$$

$$\text{RefLayerRegionBottomOffset} = \text{ref_region_bottom_offset}[\text{rLId}] * \text{SubHeightRefLayerC} \quad (\text{H-5})$$

The variables `RefLayerRegionWidthInSamplesY` and `RefLayerRegionHeightInSamplesY` are the width and height of the reference region in the decoded direct reference layer picture `rlPic` in units of luma samples, respectively, and are derived as follows:

$$\text{RefLayerRegionWidthInSamplesY} = \text{PicWidthInSamplesRefLayerY} - \text{RefLayerRegionLeftOffset} - \text{RefLayerRegionRightOffset} \quad (\text{H-6})$$

$$\text{RefLayerRegionHeightInSamplesY} = \text{PicHeightInSamplesRefLayerY} - \text{RefLayerRegionTopOffset} - \text{RefLayerRegionBottomOffset} \quad (\text{H-7})$$

The variables `PicWidthInSamplesCurrC`, `PicHeightInSamplesCurrC`, `PicWidthInSamplesRefLayerC` and `PicHeightInSamplesRefLayerC` are derived as follows:

$$\text{PicWidthInSamplesCurrC} = \text{PicWidthInSamplesCurrY} / \text{SubWidthCurrC} \quad (\text{H-8})$$

$$\text{PicHeightInSamplesCurrC} = \text{PicHeightInSamplesCurrY} / \text{SubHeightCurrC} \quad (\text{H-9})$$

$$\text{PicWidthInSamplesRefLayerC} = \text{PicWidthInSamplesRefLayerY} / \text{SubWidthRefLayerC} \quad (\text{H-10})$$

$$\text{PicHeightInSamplesRefLayerC} = \text{PicHeightInSamplesRefLayerY} / \text{SubHeightRefLayerC} \quad (\text{H-11})$$

The variables `ScaledRefLayerLeftOffset`, `ScaledRefLayerTopOffset`, `ScaledRefLayerRightOffset` and `ScaledRefLayerBottomOffset` are derived as follows:

$$\text{ScaledRefLayerLeftOffset} = \text{scaled_ref_layer_left_offset}[\text{rLId}] * \text{SubWidthCurrC} \quad (\text{H-12})$$

$$\text{ScaledRefLayerTopOffset} = \text{scaled_ref_layer_top_offset}[\text{rLId}] * \text{SubHeightCurrC} \quad (\text{H-13})$$

$$\text{ScaledRefLayerRightOffset} = \text{scaled_ref_layer_right_offset}[\text{rLId}] * \text{SubWidthCurrC} \quad (\text{H-14})$$

$$\text{ScaledRefLayerBottomOffset} = \text{scaled_ref_layer_bottom_offset}[\text{rLId}] * \text{SubHeightCurrC} \quad (\text{H-15})$$

The variables `ScaledRefRegionWidthInSamplesY` and `ScaledRefRegionHeightInSamplesY` are derived as follows:

$$\text{ScaledRefRegionWidthInSamplesY} = \text{PicWidthInSamplesCurrY} - \text{ScaledRefLayerLeftOffset} - \text{ScaledRefLayerRightOffset} \quad (\text{H-16})$$

$$\text{ScaledRefRegionHeightInSamplesY} = \text{PicHeightInSamplesCurrY} - \text{ScaledRefLayerTopOffset} - \text{ScaledRefLayerBottomOffset} \quad (\text{H-17})$$

The variables $\text{SpatialScaleFactorHorY}$, $\text{SpatialScaleFactorVerY}$, $\text{SpatialScaleFactorHorC}$ and $\text{SpatialScaleFactorVerC}$ are derived as follows:

$$\text{SpatialScaleFactorHorY} = ((\text{RefLayerRegionWidthInSamplesY} \ll 16) + (\text{ScaledRefRegionWidthInSamplesY} \gg 1)) / \text{ScaledRefRegionWidthInSamplesY} \quad (\text{H-18})$$

$$\text{SpatialScaleFactorVerY} = ((\text{RefLayerRegionHeightInSamplesY} \ll 16) + (\text{ScaledRefRegionHeightInSamplesY} \gg 1)) / \text{ScaledRefRegionHeightInSamplesY} \quad (\text{H-19})$$

$$\text{SpatialScaleFactorHorC} = (((\text{RefLayerRegionWidthInSamplesY} / \text{SubWidthRefLayerC}) \ll 16) + ((\text{ScaledRefRegionWidthInSamplesY} / \text{SubWidthCurrC}) \gg 1)) / (\text{ScaledRefRegionWidthInSamplesY} / \text{SubWidthCurrC}) \quad (\text{H-20})$$

$$\begin{aligned} \text{SpatialScaleFactorVerC} = & (((\text{RefLayerRegionHeightInSamplesY} / \text{SubHeightRefLayerC}) \ll 16) + \\ & ((\text{ScaledRefRegionHeightInSamplesY} / \text{SubHeightCurrC}) \gg 1)) / (\text{ScaledRefRegionHeightInSamplesY} / \text{SubHeightCurrC}) \end{aligned} \quad (\text{H-21})$$

The variables PhaseHorY , PhaseVerY , PhaseHorC and PhaseVerC are set as follows:

$$\text{PhaseHorY} = \text{phase_hor_luma}[\text{rLId}] \quad (\text{H-22})$$

$$\text{PhaseVerY} = \text{phase_ver_luma}[\text{rLId}] \quad (\text{H-23})$$

$$\text{PhaseHorC} = \text{phase_hor_chroma_plus8}[\text{rLId}] - 8 \quad (\text{H-24})$$

$$\text{PhaseVerC} = \text{phase_ver_chroma_plus8}[\text{rLId}] - 8 \quad (\text{H-25})$$

It is a requirement of bitstream conformance that BitDepthRefLayerY shall be less than or equal to BitDepthCurrY and BitDepthRefLayerC shall be less than or equal to BitDepthCurrC .

It is a requirement of bitstream conformance that the values of $\text{RefLayerRegionWidthInSamplesY}$, $\text{RefLayerRegionHeightInSamplesY}$, $\text{ScaledRefRegionWidthInSamplesY}$ and $\text{ScaledRefRegionHeightInSamplesY}$ shall be greater than 0.

It is a requirement of bitstream conformance that $\text{ScaledRefRegionWidthInSamplesY}$ shall be greater than or equal to $\text{RefLayerRegionWidthInSamplesY}$ and $\text{ScaledRefRegionHeightInSamplesY}$ shall be greater than or equal to $\text{RefLayerRegionHeightInSamplesY}$.

It is a requirement of bitstream conformance that, when $\text{ScaledRefRegionWidthInSamplesY}$ is equal to $\text{RefLayerRegionWidthInSamplesY}$, PhaseHorY shall be equal to 0, when $\text{ScaledRefRegionWidthInSamplesC}$ is equal to $\text{RefLayerRegionWidthInSamplesC}$, PhaseHorC shall be equal to 0, when $\text{ScaledRefRegionHeightInSamplesY}$ is equal to $\text{RefLayerRegionHeightInSamplesY}$, PhaseVerY shall be equal to 0, and when $\text{ScaledRefRegionHeightInSamplesC}$ is equal to $\text{RefLayerRegionHeightInSamplesC}$, PhaseVerC shall be equal to 0.

The inter-layer reference picture ilRefPic is derived as follows:

- The variables $\text{sampleProcessingFlag}$ and $\text{motionProcessingFlag}$ are initialized to 0.
- The variable $\text{equalPictureSizeAndOffsetFlag}$ is derived as follows:
 - If all of the following conditions are true, $\text{equalPictureSizeAndOffsetFlag}$ is set equal to 1:

- PicWidthInSamplesCurrY is equal to PicWidthInSamplesRefLayerY
- PicHeightInSamplesCurrY is equal to PicHeightInSamplesRefLayerY
- ScaledRefLayerLeftOffset is equal to RefLayerRegionLeftOffset
- ScaledRefLayerTopOffset is equal to RefLayerRegionTopOffset
- ScaledRefLayerRightOffset is equal to RefLayerRegionRightOffset
- ScaledRefLayerBottomOffset is equal to RefLayerRegionBottomOffset
- PhaseHorY, PhaseVerY, PhaseHorC and PhaseVerC are all equal to 0
- Otherwise, equalPictureSizeAndOffsetFlag is set equal to 0.
- The variable currColourMappingEnableFlag is derived as follows:
 - If colour_mapping_enabled_flag is equal to 1 and if there exists a value of i, with i in the range of 0 to num_cm_ref_layers_minus1, inclusive, for which cm_ref_layer_id[i] is equal to rLId, currColourMappingEnableFlag is set equal to 1.
 - Otherwise, currColourMappingEnableFlag is set equal to 0.
- If equalPictureSizeAndOffsetFlag is equal to 1, BitDepthRefLayerY is equal to BitDepthCurrY, BitDepthRefLayerC is equal to BitDepthCurrC, SubWidthRefLayerC is equal to SubWidthCurrC, SubHeightRefLayerC is equal to SubHeightCurrC and currColourMappingEnableFlag is equal to 0, ilRefPic is set equal to rLPic.
- Otherwise, the following applies:
 - The inter-layer reference picture ilRefPic is generated as follows:
 - The PicOrderCntVal value of ilRefPic is set equal to the PicOrderCntVal value of rLPic.
 - The nuh_layer_id value of ilRefPic is set equal to rLId.
 - The variable currLayerId is set equal to the value of nuh_layer_id of the current picture.
 - When VpsInterLayerSamplePredictionEnabled[LayerIdxInVps[currLayerId]][LayerIdxInVps[rLId]] is equal to 1, the following applies:
 - If currColourMappingEnableFlag is equal to 1, the following applies:
 - The colour mapping process as specified in clause H.8.1.4.4 is invoked with the picture sample arrays, rLPicSample_L, rLPicSample_{Cb} and rLPicSample_{Cr}, of the direct reference layer picture rLPic as inputs, and with the colour mapped picture sample arrays, cmPicSample_L, cmPicSample_{Cb} and cmPicSample_{Cr} of the colour mapped reference layer picture cmPic as outputs.
 - BitDepthRefLayerY and BitDepthRefLayerC are set equal to BitDepthCmOutputY and BitDepthCmOutputC, respectively.
 - If equalPictureSizeAndOffsetFlag is equal to 1, BitDepthRefLayerY is equal to BitDepthCurrY, BitDepthRefLayerC is equal to BitDepthCurrC, SubWidthRefLayerC is equal to SubWidthCurrC and SubHeightRefLayerC is equal to SubHeightCurrC, the picture sample arrays rsPicSample_L, rsPicSample_{Cb} and rsPicSample_{Cr} of the inter-layer reference picture ilRefPic are set equal to cmPicSample_L, cmPicSample_{Cb} and cmPicSample_{Cr}, respectively.
 - Otherwise, the picture sample resampling process as specified in clause H.8.1.4.2 is invoked with the picture sample arrays, cmPicSample_L, cmPicSample_{Cb} and cmPicSample_{Cr}, of the colour mapped reference layer picture cmPic as inputs, and with the resampled picture sample arrays, rsPicSample_L, rsPicSample_{Cb} and rsPicSample_{Cr} of the inter-layer reference picture ilRefPic as outputs.
 - Otherwise, the picture sample resampling process as specified in clause H.8.1.4.2 is invoked with the picture sample arrays, rLPicSample_L, rLPicSample_{Cb} and rLPicSample_{Cr}, of the direct reference layer picture rLPic as inputs, and with the resampled picture sample arrays, rsPicSample_L, rsPicSample_{Cb} and rsPicSample_{Cr} of the inter-layer reference picture ilRefPic as outputs.
 - sampleProcessingFlag is set equal to 1.
 - When VpsInterLayerMotionPredictionEnabled[LayerIdxInVps[currLayerId]][LayerIdxInVps[rLId]] is equal to 1, the following applies:
 - A single slice ilRefSlice of the inter-layer reference picture ilRefPic is generated as follows:

- The values of slice_type, num_ref_idx_l0_active_minus1 and num_ref_idx_l1_active_minus1 for the generated slice ilRefSlice are set equal to the values of slice_type, num_ref_idx_l0_active_minus1 and num_ref_idx_l1_active_minus1, respectively, of the first slice of rIPic.
- When ilRefSlice is a P or B slice, for i in the range of 0 to num_ref_idx_l0_active_minus1 of ilRefSlice, inclusive, the reference picture associated with index i in reference picture list 0 of ilRefSlice is set equal to the reference picture associated with index i in reference picture list 0 of the first slice of rIPic.
- When ilRefSlice is a B slice, for i in the range of 0 to num_ref_idx_l1_active_minus1 of ilRefSlice, inclusive, the reference picture associated with index i in reference picture list 1 of ilRefSlice is set equal to the reference picture associated with index i in reference picture list 1 of the first slice of rIPic.
 NOTE – When the inter-layer reference picture ilRefPic is used as the collocated picture for temporal motion vector prediction, all slices of rIPic are constrained to have the same values of slice_type, num_ref_idx_l0_active_minus1 and num_ref_idx_l1_active_minus1.
- If equalPictureSizeAndOffsetFlag is equal to 1, the following applies:
 - The motion and mode parameters of the inter-layer reference picture ilRefPic, including an array CuPredMode specifying the prediction modes, two arrays RefIdxL0 and RefIdxL1 specifying the reference indices, two arrays MvL0 and MvL1 specifying the luma motion vectors, and two arrays PredFlagL0 and PredFlagL1 specifying the prediction list utilization flags, are set equal to those of the decoded direct reference layer picture rIPic, respectively.
- Otherwise, the following applies:
 - The picture motion and mode parameters resampling process as specified in clause H.8.1.4.3 is invoked with the direct reference layer picture rIPic, an array rIPredMode specifying the prediction modes CuPredMode of rIPic, two arrays rIPrefIdxL0 and rIPrefIdxL1 specifying the reference indices of rIPic, two arrays rIMvL0 and rIMvL1 specifying the luma motion vectors of rIPic, and two arrays rIPredFlagL0 and rIPredFlagL1 specifying the prediction list utilization flags of rIPic as inputs, and an array rsPredMode specifying the prediction modes CuPredMode of ilRefPic, two arrays rsRefIdxL0 and rsRefIdxL1 specifying the reference indices of ilRefPic, two arrays rsMvL0 and rsMvL1 specifying the luma motion vectors of ilRefPic, and two arrays rsPredFlagL0 and rsPredFlagL1 specifying the prediction list utilization flags of ilRefPic as outputs.
 - motionProcessingFlag is set equal to 1.
- The following applies:

$$\text{NumRefLayerPicsSampleProcessing} += \text{sampleProcessingFlag} \quad (\text{H-26})$$

$$\text{NumRefLayerPicsMotionProcessing} += \text{motionProcessingFlag} \quad (\text{H-27})$$

$$\text{NumRefLayerPicsProcessing} += \text{sampleProcessingFlag} \mid \mid \text{motionProcessingFlag} \quad (\text{H-28})$$

H.8.1.4.2 Resampling process of picture sample values

H.8.1.4.2.1 General

Inputs to this process are:

- a (PicWidthInSamplesRefLayerY)x(PicHeightInSamplesRefLayerY) array rIPicSample_L of luma samples,
- a (PicWidthInSamplesRefLayerC)x(PicHeightInSamplesRefLayerC) array rIPicSample_{Cb} of chroma samples of the component Cb,
- a (PicWidthInSamplesRefLayerC)x(PicHeightInSamplesRefLayerC) array rIPicSample_{Cr} of chroma samples of the component Cr.

Outputs of this process are:

- a (PicWidthInSamplesCurrY)x(PicHeightInSamplesCurrY) array rsPicSample_L of luma samples,
- a (PicWidthInSamplesCurrC)x(PicHeightInSamplesCurrC) array rsPicSample_{Cb} of chroma samples of the component Cb,

- a $(\text{PicWidthInSamplesCurrC}) \times (\text{PicHeightInSamplesCurrC})$ array rsPicSample_{Cr} of chroma samples of the component Cr .

The resampled luma sample value $\text{rsPicSample}_L[xP][yP]$, with $xP = 0..\text{PicWidthInSamplesCurrY} - 1$, $yP = 0..\text{PicHeightInSamplesCurrY} - 1$, is derived by invoking the luma sample resampling process specified in clause H.8.1.4.2.2 with luma sample location (xP, yP) and the reference luma sample array rPicSample_L given as inputs.

The resampled chroma sample value $\text{rsPicSample}_{Cb}[xP_C][yP_C]$, with $xP_C = 0..\text{PicWidthInSamplesCurrC} - 1$, $yP_C = 0..\text{PicHeightInSamplesCurrC} - 1$, of the chroma component Cb is derived by invoking the chroma sample resampling process specified in clause H.8.1.4.2.3 with chroma sample location (xP_C, yP_C) and the reference chroma sample array rPicSample_{Cb} given as inputs.

The resampled chroma sample value $\text{rsPicSample}_{Cr}[xP_C][yP_C]$, with $xP_C = 0..\text{PicWidthInSamplesCurrC} - 1$, $yP_C = 0..\text{PicHeightInSamplesCurrC} - 1$, of the chroma component Cr is derived by invoking the chroma sample resampling process specified in clause H.8.1.4.2.3 with chroma sample location (xP_C, yP_C) and the reference sample array rPicSample_{Cr} given as inputs.

H.8.1.4.2.2 Resampling process of luma sample values

Inputs to this process are

- a luma sample location (xP, yP) relative to the top-left luma sample of the current picture,
- the luma reference sample array rPicSample_L .

Output of this process is the resampled luma sample value rsLumaSample .

Table H.1 specifies the 8-tap filter coefficients $f_L[p, x]$ with $p = 0..15$ and $x = 0..7$ used for the luma resampling process.

Table H.1 – 16-phase luma resampling filter

Phase p	Interpolation filter coefficients							
	$f_L[p, 0]$	$f_L[p, 1]$	$f_L[p, 2]$	$f_L[p, 3]$	$f_L[p, 4]$	$f_L[p, 5]$	$f_L[p, 6]$	$f_L[p, 7]$
0	0	0	0	64	0	0	0	0
1	0	1	−3	63	4	−2	1	0
2	−1	2	−5	62	8	−3	1	0
3	−1	3	−8	60	13	−4	1	0
4	−1	4	−10	58	17	−5	1	0
5	−1	4	−11	52	26	−8	3	−1
6	−1	3	−9	47	31	−10	4	−1
7	−1	4	−11	45	34	−10	4	−1
8	−1	4	−11	40	40	−11	4	−1
9	−1	4	−10	34	45	−11	4	−1
10	−1	4	−10	31	47	−9	3	−1
11	−1	3	−8	26	52	−11	4	−1
12	0	1	−5	17	58	−10	4	−1
13	0	1	−4	13	60	−8	3	−1
14	0	1	−3	8	62	−5	2	−1
15	0	1	−2	4	63	−3	1	0

The value of the resampled luma sample rsLumaSample is derived by applying the following ordered steps:

1. The derivation process for reference layer sample location used in resampling as specified in clause H.8.1.4.2.4 is invoked with chromaFlag equal to 0 and luma sample location (xP, yP) given as the inputs and luma sample location $(x\text{Ref}16, y\text{Ref}16)$ in units of 1/16-th luma sample as output.
2. The variables $x\text{Ref}$ and $x\text{Phase}$ are derived as follows:

$$x\text{Ref} = (x\text{Ref}16 \gg 4) \quad (\text{H-29})$$

$$x\text{Phase} = (x\text{Ref}16) \% 16 \quad (\text{H-30})$$

3. The variables $y\text{Ref}$ and $y\text{Phase}$ are derived as follows:

$$yRef = (yRef16 \gg 4) \quad (H-31)$$

$$yPhase = (yRef16) \% 16 \quad (H-32)$$

4. The variables shift1, shift2 and offset are derived as follows:

$$shift1 = BitDepthRefLayerY - 8 \quad (H-33)$$

$$shift2 = 20 - BitDepthCurrY \quad (H-34)$$

$$offset = 1 \ll (shift2 - 1) \quad (H-35)$$

5. The sample value tempArray[n] with $n = 0..7$, is derived as follows:

$$yPosRL = Clip3(0, PicHeightInSamplesRefLayerY - 1, yRef + n - 3) \quad (H-36)$$

$$refW = PicWidthInSamplesRefLayerY \quad (H-37)$$

$$\begin{aligned} tempArray[n] = & (f_L[xPhase, 0] * rPicSample_L[Clip3(0, refW - 1, xRef - 3), yPosRL] + \\ & f_L[xPhase, 1] * rPicSample_L[Clip3(0, refW - 1, xRef - 2), yPosRL] + \\ & f_L[xPhase, 2] * rPicSample_L[Clip3(0, refW - 1, xRef - 1), yPosRL] + \\ & f_L[xPhase, 3] * rPicSample_L[Clip3(0, refW - 1, xRef), yPosRL] + \\ & f_L[xPhase, 4] * rPicSample_L[Clip3(0, refW - 1, xRef + 1), yPosRL] + \\ & f_L[xPhase, 5] * rPicSample_L[Clip3(0, refW - 1, xRef + 2), yPosRL] + \\ & f_L[xPhase, 6] * rPicSample_L[Clip3(0, refW - 1, xRef + 3), yPosRL] + \end{aligned} \quad (H-38)$$

$$f_L[xPhase, 7] * rPicSample_L[Clip3(0, refW - 1, xRef + 4), yPosRL]) \gg shift1$$

6. The resampled luma sample value rsLumaSample is derived as follows:

$$\begin{aligned} rsLumaSample = & (f_L[yPhase, 0] * tempArray[0] + \\ & f_L[yPhase, 1] * tempArray[1] + \\ & f_L[yPhase, 2] * tempArray[2] + \\ & f_L[yPhase, 3] * tempArray[3] + \\ & f_L[yPhase, 4] * tempArray[4] + \\ & f_L[yPhase, 5] * tempArray[5] + \\ & f_L[yPhase, 6] * tempArray[6] + \\ & f_L[yPhase, 7] * tempArray[7] + offset) \gg shift2 \end{aligned} \quad (H-39)$$

$$rsLumaSample = Clip3(0, (1 \ll BitDepthCurrY) - 1, rsLumaSample) \quad (H-40)$$

H.8.1.4.2.3 Resampling process of chroma sample values

Inputs to this process are:

- a chroma sample location (xP_C, yP_C) relative to the top-left chroma sample of the current picture,
- the chroma reference sample array rPicSample_C.

Output of this process is the resampled chroma sample value rsChromaSample.

Table H.2 specifies the 4-tap filter coefficients $fc[p, x]$ with $p = 0..15$ and $x = 0..3$ used for the chroma resampling process.

Table H.2 – 16-phase chroma resampling filter

Phase p	Interpolation filter coefficients			
	$fc[p, 0]$	$fc[p, 1]$	$fc[p, 2]$	$fc[p, 3]$
0	0	64	0	0
1	−2	62	4	0
2	−2	58	10	−2
3	−4	56	14	−2
4	−4	54	16	−2
5	−6	52	20	−2
6	−6	46	28	−4
7	−4	42	30	−4

8	-4	36	36	-4
9	-4	30	42	-4
10	-4	28	46	-6
11	-2	20	52	-6
12	-2	16	54	-4
13	-2	14	56	-4
14	-2	10	58	-2
15	0	4	62	-2

The value of the resampled chroma sample value rsChromaSample is derived by applying the following ordered steps:

1. The derivation process for reference layer sample location in resampling as specified in clause H.8.1.4.2.4 is invoked with chromaFlag equal to 1 and chroma sample location (xP_C, yP_C) given as the inputs and chroma sample location (xRef16, yRef16) in units of 1/16-th chroma sample as output.

2. The variables xRef and xPhase are derived as follows:

$$xRef = (xRef16 \gg 4) \quad (H-41)$$

$$xPhase = (xRef16) \% 16 \quad (H-42)$$

3. The variables yRef and yPhase are derived as follows:

$$yRef = (yRef16 \gg 4) \quad (H-43)$$

$$yPhase = (yRef16) \% 16 \quad (H-44)$$

4. The variables shift1, shift2 and offset are derived as follows:

$$shift1 = BitDepthRefLayerC - 8 \quad (H-45)$$

$$shift2 = 20 - BitDepthCurrC \quad (H-46)$$

$$offset = 1 \ll (shift2 - 1) \quad (H-47)$$

5. The sample value tempArray[n] with n = 0..3, is derived as follows:

$$yPosRL = Clip3(0, PicHeightInSamplesRefLayerC - 1, yRef + n - 1) \quad (H-48)$$

$$refWC = PicWidthInSamplesRefLayerC \quad (H-49)$$

$$tempArray[n] = (f_c[xPhase, 0] * rPicSamplec[Clip3(0, refWC - 1, xRef - 1), yPosRL] + f_c[xPhase, 1] * rPicSamplec[Clip3(0, refWC - 1, xRef), yPosRL] + f_c[xPhase, 2] * rPicSamplec[Clip3(0, refWC - 1, xRef + 1), yPosRL] + \quad (H-50)$$

$$f_c[xPhase, 3] * rPicSamplec[Clip3(0, refWC - 1, xRef + 2), yPosRL]) \gg shift1$$

6. The resampled chroma sample value rsChromaSample is derived as follows:

$$rsChromaSample = (f_c[yPhase, 0] * tempArray[0] + f_c[yPhase, 1] * tempArray[1] + f_c[yPhase, 2] * tempArray[2] + f_c[yPhase, 3] * tempArray[3] + offset) \gg shift2 \quad (H-51)$$

$$rsChromaSample = Clip3(0, (1 \ll BitDepthCurrC) - 1, rsChromaSample) \quad (H-52)$$

H.8.1.4.2.4 Derivation process for reference layer sample location in units of 1/16-th sample

Inputs to this process are:

- a variable chromaFlag specifying whether the sample location being derived is that of the luma component or that of one of the chroma colour components.
- a sample location (xP, yP) relative to the top-left sample of the luma component or the top-left sample of one of the chroma components of the current picture depending on the value of chromaFlag.

Output of this process is a sample location (xRef16, yRef16) specifying the reference layer sample location in units of 1/16-th sample relative to the top-left sample of the luma component or the top-left sample of one of the chroma components of the direct reference layer picture depending on the value of chromaFlag.

The variables currOffsetLeft, currOffsetTop, refOffsetLeft and refOffsetTop are derived as follows:

$$\text{currOffsetLeft} = \text{ScaledRefLayerLeftOffset} / (\text{chromaFlag} ? \text{SubWidthCurrC} : 1) \quad (\text{H-53})$$

$$\text{currOffsetTop} = \text{ScaledRefLayerTopOffset} / (\text{chromaFlag} ? \text{SubHeightCurrC} : 1) \quad (\text{H-54})$$

$$\text{refOffsetLeft} = (\text{RefLayerRegionLeftOffset} / (\text{chromaFlag} ? \text{SubWidthRefLayerC} : 1)) \ll 4 \quad (\text{H-55})$$

$$\text{refOffsetTop} = (\text{RefLayerRegionTopOffset} / (\text{chromaFlag} ? \text{SubHeightRefLayerC} : 1)) \ll 4 \quad (\text{H-56})$$

The variables phaseHor, phaseVer, scaleHor and scaleVer are derived as follows:

$$\text{phaseHor} = \text{chromaFlag} ? \text{PhaseHorC} : \text{PhaseHorY} \quad (\text{H-57})$$

$$\text{phaseVer} = \text{chromaFlag} ? \text{PhaseVerC} : \text{PhaseVerY} \quad (\text{H-58})$$

$$\text{scaleHor} = \text{chromaFlag} ? \text{SpatialScaleFactorHorC} : \text{SpatialScaleFactorHorY} \quad (\text{H-59})$$

$$\text{scaleVer} = \text{chromaFlag} ? \text{SpatialScaleFactorVerC} : \text{SpatialScaleFactorVerY} \quad (\text{H-60})$$

The variables addHor and addVer are derived as follows:

$$\text{addHor} = - ((\text{scaleHor} * \text{phaseHor} + 8) \gg 4) \quad (\text{H-61})$$

$$\text{addVer} = - ((\text{scaleVer} * \text{phaseVer} + 8) \gg 4) \quad (\text{H-62})$$

The variables xRef16 and yRef16 are derived as follows:

$$\text{xRef16} = (((\text{xP} - \text{currOffsetLeft}) * \text{scaleHor} + \text{addHor} + (1 \ll 11)) \gg 12) + \text{refOffsetLeft} \quad (\text{H-63})$$

$$\text{yRef16} = (((\text{yP} - \text{currOffsetTop}) * \text{scaleVer} + \text{addVer} + (1 \ll 11)) \gg 12) + \text{refOffsetTop} \quad (\text{H-64})$$

H.8.1.4.3 Resampling process of picture motion and mode parameters

Inputs to this process are:

- a decoded direct reference layer picture rIPic,
- a (PicWidthInSamplesRefLayerY)x(PicHeightInSamplesRefLayerY) array rIPredMode specifying the prediction modes of the direct reference layer picture,
- two (PicWidthInSamplesRefLayerY)x(PicHeightInSamplesRefLayerY) arrays rIRefIdxL0 and rIRefIdxL1 specifying the reference indices of the direct reference layer picture,
- two (PicWidthInSamplesRefLayerY)x(PicHeightInSamplesRefLayerY) arrays rIMvL0 and rIMvL1 specifying the luma motion vectors of the direct reference layer picture,
- two (PicWidthInSamplesRefLayerY)x(PicHeightInSamplesRefLayerY) arrays rIPredFlagL0 and rIPredFlagL1 specifying the prediction list utilization flags of the direct reference layer picture.

Outputs of this process are:

- a (PicWidthInSamplesCurrY)x(PicHeightInSamplesCurrY) array rsPredMode specifying the prediction modes of the resampled picture,

- two (PicWidthInSamplesCurrY)x(PicHeightInSamplesCurrY) arrays rsRefIdxL0 and rsRefIdxL1 specifying the reference indices of the resampled picture,
- two (PicWidthInSamplesCurrY)x(PicHeightInSamplesCurrY) arrays rsMvL0 and rsMvL1 specifying the luma motion vectors of the resampled picture,
- two (PicWidthInSamplesCurrY)x(PicHeightInSamplesCurrY) arrays rsPredFlagL0 and rsPredFlagL1 specifying the prediction list utilization flags of the resampled picture.

For each 16x16 prediction block of the resampled picture with the top-left luma sample location at (xP, yP), where $xP = xB \ll 4$ and $yP = yB \ll 4$, for $xB = 0..((PicWidthInSamplesCurrY + 15) \gg 4) - 1$ and $yB = 0..((PicHeightInSamplesCurrY + 15) \gg 4) - 1$, its motion and mode parameters rsPredMode[xP][yP], rsMvLX[xP][yP], rsRefIdxLX[xP][yP] and rsPredFlagLX[xP][yP], with X being equal to 0 and 1, are derived by applying the following ordered steps:

1. The centre location (xPCtr, yPCtr) of the luma prediction block is derived as follows:

$$xPCtr = xP + 8 \quad (H-65)$$

$$yPCtr = yP + 8 \quad (H-66)$$

2. The variables xRef and yRef are derived as follows:

$$\begin{aligned} xRef = & \\ & (((xPCtr - ScaledRefLayerLeftOffset) * SpatialScaleFactorHorY + (1 \ll 15)) \gg 16) \\ & + RefLayerRegionLeftOffset \end{aligned} \quad (H-67)$$

$$\begin{aligned} yRef = & \\ & (((yPCtr - ScaledRefLayerTopOffset) * SpatialScaleFactorVerY + (1 \ll 15)) \gg 16) \\ & + RefLayerRegionTopOffset \end{aligned} \quad (H-68)$$

3. The rounded reference layer luma sample location (xRL, yRL) is derived as follows:

$$xRL = ((xRef + 4) \gg 4) \ll 4 \quad (H-69)$$

$$yRL = ((yRef + 4) \gg 4) \ll 4 \quad (H-70)$$

4. The variable rsPredMode[xP][yP] is derived as follows:

$$\begin{aligned} & \text{if}(xRL < 0 \mid \mid xRL \geq PicWidthInSamplesRefLayerY \mid \mid yRL < 0 \mid \mid \\ & \quad yRL \geq PicHeightInSamplesRefLayerY), \quad (H-71) \\ & \quad rsPredMode[xP][yP] = MODE_INTRA \\ & \text{else} \\ & \quad rsPredMode[xP][yP] = rlPredMode[xRL][yRL] \end{aligned}$$

5. For X being each of 0 and 1, the variables rsMvLX[xP][yP], rsRefIdxLX[xP][yP] and rsPredFlagLX[xP][yP] are derived as follows:

- If rsPredMode[xP][yP] is equal to MODE_INTER, the following applies:

- rsRefIdxLX[xP][yP] and rsPredFlagLX[xP][yP] are derived as follows:

$$rsRefIdxLX[xP][yP] = rlRefIdxLX[xRL][yRL] \quad (H-72)$$

$$rsPredFlagLX[xP][yP] = rlPredFlagLX[xRL][yRL] \quad (H-73)$$

- rsMvLX[xP][yP][0] is derived as follows:

- If ScaledRefRegionWidthInSamplesY is not equal to RefLayerRegionWidthInSamplesY, the following applies:

$$\text{scaleMVX} = \text{Clip3}(-4096, 4095, ((\text{ScaledRefRegionWidthInSamplesY} \ll 8)(\text{H-74}) + (\text{RefLayerRegionWidthInSamplesY} \gg 1)) / \text{RefLayerRegionWidthInSamplesY})$$

$$\begin{aligned} \text{rsMvLX}[xP][yP][0] &= \text{Clip3}(-32768, 32767, \text{Sign}(\text{scaleMVX} * \\ &\quad \text{rlMvLX}[xRL][yRL][0]) * ((\text{Abs}(\text{scaleMVX} * \text{rlMvLX}[xRL][yRL][0]) \\ &\quad (\text{H-75}) \\ &\quad + 127) \gg 8)) \end{aligned}$$

- Otherwise, the following applies:

$$\text{rsMvLX}[xP][yP][0] = \text{rlMvLX}[xRL][yRL][0] \quad (\text{H-76})$$

- rsMvLX[xP][yP][1] is derived as follows:

- If ScaledRefRegionHeightInSamplesY is not equal to RefLayerRegionHeightInSamplesY, the following applies:

$$\begin{aligned} \text{scaleMVY} &= \text{Clip3}(-4096, 4095, ((\text{ScaledRefRegionHeightInSamplesY} \ll 8)(\text{H-77}) \\ &\quad + (\text{RefLayerRegionHeightInSamplesY} \gg 1)) / \\ &\quad \text{RefLayerRegionHeightInSamplesY}) \end{aligned}$$

$$\begin{aligned} \text{rsMvLX}[xP][yP][1] &= \text{Clip3}(-32768, 32767, \text{Sign}(\text{scaleMVY} * \\ &\quad \text{rlMvLX}[xRL][yRL][1]) * ((\text{Abs}(\text{scaleMVY} * \text{rlMvLX}[xRL][yRL][1]) \\ &\quad (\text{H-78}) \\ &\quad + 127) \gg 8)) \end{aligned}$$

- Otherwise, the following applies:

$$\text{rsMvLX}[xP][yP][1] = \text{rlMvLX}[xRL][yRL][1] \quad (\text{H-79})$$

- Otherwise (rsPredMode[xP][yP] is equal to MODE_INTRA), the following applies:
 - rsMvL0[xP][yP][0], rsMvL0[xP][yP][1], rsMvL1[xP][yP][0] and rsMvL1[xP][yP][1] are set equal to 0.
 - rsRefIdxL0[xP][yP] and rsRefIdxL1[xP][yP] are set equal to -1.
 - rsPredFlagL0[xP][yP] and rsPredFlagL1[xP][yP] are set equal to 0.

H.8.1.4.4 Colour mapping process of picture sample values

H.8.1.4.4.1 General

Inputs to this process are:

- a (PicWidthInSamplesRefLayerY)x(PicHeightInSamplesRefLayerY) array rlPicSample_L of luma samples,
- a (PicWidthInSamplesRefLayerC)x(PicHeightInSamplesRefLayerC) array rlPicSample_{Cb} of chroma samples of the component Cb,
- a (PicWidthInSamplesRefLayerC)x(PicHeightInSamplesRefLayerC) array rlPicSample_{Cr} of chroma samples of the component Cr.

Outputs of this process are:

- a (PicWidthInSamplesRefLayerY)x(PicHeightInSamplesRefLayerY) array cmPicSample_L of luma samples,
- a (PicWidthInSamplesRefLayerC)x(PicHeightInSamplesRefLayerC) array cmPicSample_{Cb} of chroma samples of the component Cb,
- a (PicWidthInSamplesRefLayerC)x(PicHeightInSamplesRefLayerC) array cmPicSample_{Cr} of chroma samples of the component Cr.

The colour mapped luma sample cmPicSample_L[xP][yP] with xP = 0..PicWidthInSamplesRefLayerY - 1, yP = 0..PicHeightInSamplesRefLayerY - 1 is derived by invoking the colour mapping process of luma sample values as

specified in clause H.8.1.4.4.2 with luma sample location (x_P, y_P), sample arrays $rlPicSampleL$, $rlPicSampleCb$ and $rlPicSampleCr$ given as inputs.

The colour mapped chroma sample $cmPicSampleCb[x_C][y_C]$ with $x_C = 0..PicWidthInSamplesRefLayerC - 1$, $y_C = 0..PicHeightInSamplesRefLayerC - 1$ is derived by invoking the colour mapping process of chroma sample values as specified in clause H.8.1.4.4.3 with chroma sample location (x_C, y_C), sample arrays $rlPicSampleL$, $rlPicSampleCb$ $rlPicSampleCr$ and $cIdx$ equal to 0 given as inputs.

The colour mapped chroma sample $cmPicSampleCr[x_C][y_C]$ with $x_C = 0..PicWidthInSamplesRefLayerC - 1$, $y_C = 0..PicHeightInSamplesRefLayerC - 1$ is derived by invoking the colour mapping process of chroma sample values as specified in clause H.8.1.4.4.3 with chroma sample location (x_C, y_C), sample arrays $rlPicSampleL$, $rlPicSampleCb$, $rlPicSampleCr$ and $cIdx$ equal to 1 given as inputs.

H.8.1.4.4.2 Colour mapping process of luma sample values

Inputs to this process are:

- a luma sample location (x_P, y_P) specifying the luma sample location relative to the top-left luma sample of the direct reference layer picture,
- the luma reference layer sample array $rlPicSample_Y$
- the chroma reference layer sample array $rlPicSample_{Cb}$ of the Cb component
- the chroma reference layer sample array $rlPicSample_{Cr}$ of the Cr component

Output of the process is a colour mapped luma sample value $cmLumaSample$

The chroma sample location (x_{P_C}, y_{P_C}) is set equal to ($x_P / SubWidthRefLayerC, y_P / SubHeightRefLayerC$).

The value of $cmLumaSample$ is derived by applying the following ordered steps:

1. The variables $yShift2Idx$, $cShift2Idx$ are derived as follows:

$$yShift2Idx = BitDepthCmInputY - cm_octant_depth - cm_y_part_num_log2 \quad (H-80)$$

$$cShift2Idx = BitDepthCmInputC - cm_octant_depth \quad (H-81)$$

2. The variables $nMappingShift$ and $nMappingOffset$ are derived as follows:

$$nMappingShift = 10 + BitDepthCmInputY - BitDepthCmOutputY \quad (H-82)$$

$$nMappingOffset = 1 \ll (nMappingShift - 1) \quad (H-83)$$

3. The variables $tempCb$ and $tempCr$ are derived as follows:

- If $SubWidthRefLayerC$ is equal to 2 and $SubHeightRefLayerC$ is equal to 2, the following applies:

- If $x_P \% 2$ is equal to 0 and $y_P \% 2$ is equal to 0, the following applies:

$$yP2_C = \text{Max}(0, y_{P_C} - 1) \quad (H-84)$$

$$tempCb = (rlPicSample_{Cb}[x_{P_C}][y_{P_C}] * 3 + rlPicSample_{Cb}[x_{P_C}][yP2_C] + 2) \gg 2 \quad (H-85)$$

$$tempCr = (rlPicSample_{Cr}[x_{P_C}][y_{P_C}] * 3 + rlPicSample_{Cr}[x_{P_C}][yP2_C] + 2) \gg 2 \quad (H-86)$$

- Otherwise, if $x_P \% 2$ is equal to 0 and $y_P \% 2$ is equal to 1, the following applies:

$$yP2_C = \text{Min}(y_{P_C} + 1, PicHeightInSamplesRefLayerC - 1) \quad (H-87)$$

$$tempCb = (rlPicSample_{Cb}[x_{P_C}][y_{P_C}] * 3 + rlPicSample_{Cb}[x_{P_C}][yP2_C] + 2) \gg 2 \quad (H-88)$$

$$tempCr = (rlPicSample_{Cr}[x_{P_C}][y_{P_C}] * 3 + rlPicSample_{Cr}[x_{P_C}][yP2_C] + 2) \gg 2 \quad (H-89)$$

- Otherwise, if $xP \% 2$ is equal to 1 and $yP \% 2$ is equal to 0, the following applies:

$$xP2_C = \text{Min}(xP_C + 1, \text{PicWidthInSamplesRefLayerC} - 1) \quad (\text{H-90})$$

$$yP2_C = \text{Max}(0, yP_C - 1) \quad (\text{H-91})$$

$$\text{tempCb} = (\text{rPicSampleCb}[xP_C][yP2_C] + \text{rPicSampleCb}[xP2_C][yP2_C] + (\text{rPicSampleCb}[xP_C][yP_C] + \text{rPicSampleCb}[xP2_C][yP_C]) * 3 + 4) \gg 3 \quad (\text{H-92})$$

$$\text{tempCr} = (\text{rPicSampleCr}[xP_C][yP2_C] + \text{rPicSampleCr}[xP2_C][yP2_C] + (\text{rPicSampleCr}[xP_C][yP_C] + \text{rPicSampleCr}[xP2_C][yP_C]) * 3 + 4) \gg 3 \quad (\text{H-93})$$
- Otherwise ($xP \% 2$ is equal to 1 and $yP \% 2$ is equal to 1), the following applies:

$$xP2_C = \text{Min}(xP_C + 1, \text{PicWidthInSamplesRefLayerC} - 1) \quad (\text{H-94})$$

$$yP2_C = \text{Min}(yP_C + 1, \text{PicHeightInSamplesRefLayerC} - 1) \quad (\text{H-95})$$

$$\text{tempCb} = ((\text{rPicSampleCb}[xP_C][yP_C] + \text{rPicSampleCb}[xP2_C][yP_C]) * 3 + \text{rPicSampleCb}[xP_C][yP2_C] + \text{rPicSampleCb}[xP2_C][yP2_C] + 4) \gg 3 \quad (\text{H-96})$$

$$\text{tempCr} = ((\text{rPicSampleCr}[xP_C][yP_C] + \text{rPicSampleCr}[xP2_C][yP_C]) * 3 + \text{rPicSampleCr}[xP_C][yP2_C] + \text{rPicSampleCr}[xP2_C][yP2_C] + 4) \gg 3 \quad (\text{H-97})$$
- Otherwise, if SubWidthRefLayerC is equal to 2, the following applies:
 - If $xP \% 2$ is equal to 1, the following applies:

$$xP2_C = \text{Min}(xP_C + 1, \text{PicWidthInSamplesRefLayerC} - 1) \quad (\text{H-98})$$

$$\text{tempCb} = (\text{rPicSampleCb}[xP_C][yP_C] + \text{rPicSampleCb}[xP2_C][yP_C] + 1) \gg 1 \quad (\text{H-99})$$

$$\text{tempCr} = (\text{rPicSampleCr}[xP_C][yP_C] + \text{rPicSampleCr}[xP2_C][yP_C] + 1) \gg 1 \quad (\text{H-100})$$
 - Otherwise ($xP \% 2$ is equal to 0), the following applies:

$$\text{tempCb} = \text{rPicSampleCb}[xP_C][yP_C] \quad (\text{H-101})$$

$$\text{tempCr} = \text{rPicSampleCr}[xP_C][yP_C] \quad (\text{H-102})$$
- Otherwise (SubWidthRefLayerC is equal to 1 and $\text{SubHeightRefLayerC}$ is equal to 1), the following applies:

$$\text{tempCb} = \text{rPicSampleCb}[xP_C][yP_C] \quad (\text{H-103})$$

$$\text{tempCr} = \text{rPicSampleCr}[xP_C][yP_C] \quad (\text{H-104})$$

4. The value of cmLumaSample is derived as follows:

$$\text{idxY} = \text{rPicSampleY}[xP][yP] \gg y\text{Shift2Idx} \quad (\text{H-105})$$

$$\text{idxCb} = (\text{cm_octant_depth} == 1) ? (\text{tempCb} \geq \text{CMThreshU}) : (\text{tempCb} \gg c\text{Shift2Idx}) \quad (\text{H-106})$$

$$\text{idxCr} = (\text{cm_octant_depth} == 1) ? (\text{tempCr} \geq \text{CMThreshV}) : (\text{tempCr} \gg c\text{Shift2Idx}) \quad (\text{H-107})$$

$$\begin{aligned} \text{cmLumaSample} = & ((\text{LutY}[\text{idxY}][\text{idxCb}][\text{idxCr}][0] * \text{rPicSample}_Y[\text{xP}][\text{yP}] \\ & + \text{LutY}[\text{idxY}][\text{idxCb}][\text{idxCr}][1] * \text{tempCb} + \text{LutY}[\text{idxY}][\text{idxCb}][\text{idxCr}][2] * \text{tempCr} \\ & + \text{nMappingOffset}) >> \text{nMappingShift}) + \text{LutY}[\text{idxY}][\text{idxCb}][\text{idxCr}][3] \end{aligned} \quad (\text{H-108})$$

$$\text{cmLumaSample} = \text{Clip3}(0, (1 << \text{BitDepthCmOutputY}) - 1, \text{cmLumaSample}) \quad (\text{H-109})$$

H.8.1.4.4.3 Colour mapping process of chroma sample values

Inputs to this process are:

- a chroma sample location (x_{P_C} , y_{P_C}) specifying the chroma sample location relative to the top-left chroma sample of the direct reference layer picture,
- the luma reference sample array rPicSample_Y ,
- the chroma reference sample array rPicSample_{Cb} of the Cb component,
- the chroma reference sample array rPicSample_{Cr} of the Cr component,
- a variable $cIdx$ specifying the chroma component index.

Output of the process is a colour mapped chroma sample value cmChromaSample .

The luma sample location (x_P , y_P) is set equal to ($x_{P_C} * \text{SubWidthRefLayerC}$, $y_{P_C} * \text{SubHeightRefLayerC}$).

The colour mapping table LutC is set equal to LutCb if $cIdx$ is equal to 0 and set equal to LutCr otherwise.

The value of cmChromaSample is derived by applying the following ordered steps:

1. The variables $yShift2Idx$, $cShift2Idx$ are derived as follows:

$$yShift2Idx = \text{BitDepthCmInputY} - \text{cm_octant_depth} - \text{cm_y_part_num_log2} \quad (\text{H-110})$$

$$cShift2Idx = \text{BitDepthCmInputC} - \text{cm_octant_depth} \quad (\text{H-111})$$

2. The variables nMappingShift and nMappingOffset are derived as follows:

$$\text{nMappingShift} = 10 + \text{BitDepthCmInputY} - \text{BitDepthCmOutputY} \quad (\text{H-112})$$

$$\text{nMappingOffset} = 1 << (\text{nMappingShift} - 1) \quad (\text{H-113})$$

3. The variable tempY is derived as follows:

- If SubWidthRefLayerC is equal to 2 and $\text{SubHeightRefLayerC}$ is equal to 2, the following applies:

$$\text{tempY} = (\text{rPicSample}_Y[\text{xP}][\text{yP}] + \text{rPicSample}_Y[\text{xP}][\text{yP} + 1] + 1) >> 1 \quad (\text{H-114})$$

- Otherwise (SubWidthRefLayerC is not equal to 2 or $\text{SubHeightRefLayerC}$ is not equal to 2), the following applies:

$$\text{tempY} = \text{rPicSample}_Y[\text{xP}][\text{yP}] \quad (\text{H-115})$$

4. The value of cmChromaSample is derived as follows:

$$\text{idxY} = \text{tempY} >> yShift2Idx \quad (\text{H-116})$$

$$\begin{aligned} \text{idxCb} = & (\text{cm_octant_depth} == 1) ? \\ & (\text{rPicSample}_{Cb}[\text{xP}_C][\text{yP}_C] >= \text{CMThreshU}) : (\text{rPicSample}_{Cb}[\text{xP}_C][\text{yP}_C] >> cShift2Idx) \end{aligned} \quad (\text{H-117})$$

$$\begin{aligned} \text{idxCr} = & (\text{cm_octant_depth} == 1) ? \\ & (\text{rPicSample}_{Cr}[\text{xP}_C][\text{yP}_C] >= \text{CMThreshV}) : (\text{rPicSample}_{Cr}[\text{xP}_C][\text{yP}_C] >> cShift2Idx) \end{aligned} \quad (\text{H-118})$$

$$\begin{aligned} \text{cmChromaSample} = & ((\text{LutC}[\text{idxY}][\text{idxCb}][\text{idxCr}][0] * \text{tempY} \\ & + \text{LutC}[\text{idxY}][\text{idxCb}][\text{idxCr}][1] * \text{rPicSampleCb}[\text{xPc}][\text{yPc}] \\ & + \text{LutC}[\text{idxY}][\text{idxCb}][\text{idxCr}][2] * \text{rPicSampleCr}[\text{xPc}][\text{yPc}] \\ & + \text{nMappingOffset}) >> \text{nMappingShift}) + \text{LutC}[\text{idxY}][\text{idxCb}][\text{idxCr}][3] \end{aligned} \quad (\text{H-119})$$

$$\text{cmChromaSample} = \text{Clip3}(0, (1 \ll \text{BitDepthCmOutputC}) - 1, \text{cmChromaSample}) \quad (\text{H-120})$$

H.8.2 NAL unit decoding process

The specification in clause F.8.2 apply.

H.8.3 Slice decoding processes

H.8.3.1 Decoding process for picture order count

The specifications in clause F.8.3.1 apply.

H.8.3.2 Decoding process for reference picture set

The specifications in clause F.8.3.2 apply.

H.8.3.3 Decoding process for generating unavailable reference pictures

The specifications in clause F.8.3.3 apply.

H.8.3.4 Decoding process for reference picture lists construction

The specifications in clause F.8.3.4 apply with the following additions:

NOTE – Because bitstreams conforming to this annex are constrained to allow only zero-valued motion vectors for inter prediction using inter-layer reference pictures, it is suggested that a scalable encoder should disable temporal motion vector prediction for the current picture (by setting `slice_temporal_mvp_enabled_flag` to zero) when the reference picture lists of all slices in the current picture include only inter-layer reference pictures. This way, the encoder would be able to avoid the need to send the slice segment header syntax elements `collocated_from_l0_flag` and `collocated_ref_idx`.

H.8.4 Decoding process for coding units coded in intra prediction mode

The specifications in clause F.8.4 apply.

H.8.5 Decoding process for coding units coded in inter prediction mode

The specifications in clause F.8.5 apply with the following additions:

It is a requirement of bitstream conformance that, for X being replaced by either 0 or 1, the variables `mvLX[0]` and `mvLX[1]` as an output of clause 8.5.3.1 shall be equal to 0 if the value of `refIdxLX` as an output of clause 8.5.3.1 corresponds to an inter-layer reference picture. That is, in any conforming bitstream, for X being replaced by either 0 or 1, upon invoking the decoding process in clause 8.5.3.1, the values of the syntax elements `merge_idx`, `mvp_lx_flag`, `ref_idx_lx`, `MvdLX` and `mvd_l1_zero_flag` shall always result in zero values for `mvLX[0]` and `mvLX[1]` when the value of `refIdxLX` of the reference picture list `RefPicListX` indicates an inter-layer reference picture.

The variable `currLayerId` is set equal to `nuh_layer_id` of the current picture.

It is a requirement of bitstream conformance that when the reference picture represented by the variable `refIdxLX` and derived by invoking clause 8.5.3.2, for X being replaced by either 0 or 1, is an inter-layer reference picture, `VpsInterLayerSamplePredictionEnabled[LayerIdxInVps[currLayerId]][LayerIdxInVps[rLid]]` shall be equal to 1, where `rLid` is set equal to `nuh_layer_id` of the direct reference layer picture from which the inter-layer reference picture is derived.

It is a requirement of bitstream conformance that when the collocated picture `colPic`, used for temporal motion vector prediction and derived by invoking clause 8.5.3.2.7, is an inter-layer reference picture, `VpsInterLayerMotionPredictionEnabled[LayerIdxInVps[currLayerId]][LayerIdxInVps[rLid]]` shall be equal to 1, where `rLid` is set equal to `nuh_layer_id` of the direct reference layer picture from which the inter-layer reference picture is derived.

It is a requirement of bitstream conformance that the collocated picture `colPic`, used for temporal motion vector prediction and derived by invoking clause 8.5.3.2.7, shall not be an inter-layer reference picture if the direct reference layer picture, from which the inter-layer reference picture is derived, is coded using two or more slice segments and any of the following conditions is true:

- The slice segment header syntax element `slice_type` of at least one of the slice segments of the direct reference layer picture is different from the slice segment header syntax element `slice_type` of another slice segment of the direct reference layer picture;
- For X being replaced by either 0 or 1, the slice segment header syntax element `num_ref_idx_lX_active_minus1` of at least one of the slice segments of the direct reference layer picture is different from the slice segment header syntax element `num_ref_idx_lX_active_minus1` of another slice segment of the direct reference layer picture;
- For X being replaced by either 0 or 1, the reference picture list `RefPicListX` of at least one of the slice segments of the direct reference layer picture is different from the reference picture list `RefPicListX` of another slice segment of the direct reference layer picture.

H.8.6 Scaling, transformation and array construction process prior to deblocking filter process

The specifications in clause F.8.6 apply.

H.8.7 In-loop filter process

The specifications in clause F.8.7 apply.

H.9 Parsing process

The specifications in clause F.9 apply.

H.10 Specification of bitstream subsets

The specifications in clause F.10 and its subclauses apply.

H.11 Profiles, tiers and levels

H.11.1 Profiles

H.11.1.1 Scalable Main and Scalable Main 10 profiles

For a layer in an output operation point associated with an OLS in a bitstream, the layer being conforming to the Scalable Main or Scalable Main 10 profile, the following applies:

- Let `olsIdx` be the OLS index of the OLS, the sub-bitstream `subBitstream` and the base layer sub-bitstream `baseBitstream` are derived as specified in clause F.11.3.

When `vps_base_layer_internal_flag` is equal to 1, the base layer sub-bitstream `baseBitstream` shall obey the following constraints:

- When the layer conforms to the Scalable Main profile, the base layer sub-bitstream `baseBitstream` shall be indicated to conform to the Main profile.
- When the layer conforms to the Scalable Main 10 profile, the base layer sub-bitstream `baseBitstream` shall be indicated to conform to the Main 10 profile or the Main profile.

The sub-bitstream `subBitstream` shall obey the following constraints:

- All active VPSs shall have `vps_num_rep_formats_minus1` in the range of 0 to 15, inclusive.
- All active SPSs for layers in `subBitstream` shall have `chroma_format_idc` equal to 1 only.
- All active SPSs for layers in `subBitstream` shall have `transform_skip_rotation_enabled_flag`, `transform_skip_context_enabled_flag`, `implicit_rdpem_enabled_flag`, `explicit_rdpem_enabled_flag`, `extended_precision_processing_flag`, `intra_smoothing_disabled_flag`, `high_precision_offsets_enabled_flag`, `persistent_rice_adaptation_enabled_flag` and `cabac_bypass_alignment_enabled_flag`, when present, equal to 0 only.
- `CtbLog2SizeY` derived from all active SPSs for layers in `subBitstream` shall be in the range of 4 to 6, inclusive.
- All active PPSs for layers in `subBitstream` shall have `log2_max_transform_skip_block_size_minus2` and `chroma_qp_offset_list_enabled_flag`, when present, equal to 0 only.
- The variables `NumRefLayerPicsProcessing`, `NumRefLayerPicsSampleProcessing` and `NumRefLayerPicsMotionProcessing` shall be less than or equal to 1 for each decoded picture with `nuh_layer_id` included in `layerIdListTarget` that was used to derive `subBitstream`.
- `ScalabilityId[j][smIdx]` derived according to any active VPS shall be equal to 0 for any `smIdx` value not equal to 2 or 3 and for any value of `j` such that `layer_id_in_nuh[j]` is among `layerIdListTarget` that was used to derive `subBitstream`.

- For a layer with `nuh_layer_id` `iNuhLid` equal to any value included in `layerIdListTarget` that was used to derive `subBitstream`, the value of `NumRefLayers[iNuhLid]`, which specifies the total number of direct and indirect reference layers and is derived as specified in F.7.4.3.1, shall be less than or equal to 4.
- All active SPSs for layers in `subBitstream` shall have `sps_range_extension_flag` and `sps_scc_extension_flag` equal to 0 only.
- All active PPSs for layers in `subBitstream` shall have `pps_range_extension_flag` and `pps_scc_extension_flag` equal to 0 only.
- When an active PPS for any layer in `subBitstream` has `tiles_enabled_flag` equal to 1, it shall have `entropy_coding_sync_enabled_flag` equal to 0.
- When an active PPS for any layer in `subBitstream` has `tiles_enabled_flag` equal to 1, `ColumnWidthInLumaSamples[i]` shall be greater than or equal to 256 for all values of `i` in the range of 0 to `num_tile_columns_minus1`, inclusive, and `RowHeightInLumaSamples[j]` shall be greater than or equal to 64 for all values of `j` in the range of 0 to `num_tile_rows_minus1`, inclusive.
- The number of times `read_bits(1)` is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing `coding_tree_unit()` data for any CTU shall be less than or equal to $5 * \text{RawCtuBits} / 3$.
- `general_level_idc` and `sub_layer_level_idc[i]` for all values of `i` in active SPSs for any layer in `subBitstream` shall not be equal to 255 (which indicates level 8.5).
- For any active VPS, `DependencyId[i]` shall be greater than `DependencyId[j]` for any values of `i` and `j` among `layerIdListTarget` that was used to derive `subBitstream` such that `AuxId[i]` is equal to `AuxId[j]` and `i` is greater than `j`.

When the layer conforms to the Scalable Main profile, the sub-bitstream `subBitstream` shall obey the following constraints:

- All active SPSs for layers in `subBitstream` shall have `bit_depth_luma_minus8` equal to 0 only.
- All active SPSs for layers in `subBitstream` shall have `bit_depth_chroma_minus8` equal to 0 only.
- All active PPSs for layers in `subBitstream` shall have `colour_mapping_enabled_flag` equal to 0 only.
- The tier and level constraints specified for the Scalable Main profile in clause H.11.2 shall be fulfilled.

When the layer conforms to the Scalable Main 10 profile, the sub-bitstream `subBitstream` shall obey the following constraints:

- All active SPSs for layers in `subBitstream` shall have `bit_depth_luma_minus8` in the range of 0 to 2, inclusive.
- All active SPSs for layers in `subBitstream` shall have `bit_depth_chroma_minus8` in the range of 0 to 2, inclusive.
- The tier and level constraints specified for the Scalable Main profile in clause H.11.2 shall be fulfilled.

In the remainder of this clause and clause H.11.2.1, all syntax elements in the `profile_tier_level()` syntax structure refer to those in the `profile_tier_level()` syntax structure associated with the layer.

Conformance of a layer in an output operation point associated with an OLS in a bitstream to the Scalable Main profile is indicated as follows:

- If `OpTid` of the output operation point is equal to `vps_max_sub_layer_minus1`, the conformance is indicated by `general_profile_idc` being equal to 7 or `general_profile_compatibility_flag[7]` being equal to 1, and `general_max_12bit_constraint_flag` being equal to 1, `general_max_10bit_constraint_flag` being equal to 1, `general_max_8bit_constraint_flag` being equal to 1, `general_max_422chroma_constraint_flag` being equal to 1, `general_max_420chroma_constraint_flag` being equal to 1, `general_max_monochrome_constraint_flag` being equal to 0, `general_intra_constraint_flag` being equal to 0, `general_one_picture_only_constraint_flag` being equal to 0 and `general_lower_bit_rate_constraint_flag` being equal to 1.
- Otherwise (`OpTid` of the output operation point is less than `vps_max_sub_layer_minus1`), the conformance is indicated by `sub_layer_profile_idc[OpTid]` being equal to 7 or `sub_layer_profile_compatibility_flag[OpTid][7]` being equal to 1, and `sub_layer_max_12bit_constraint_flag[OpTid]` being equal to 1, `sub_layer_max_10bit_constraint_flag[OpTid]` being equal to 1, `sub_layer_max_8bit_constraint_flag[OpTid]` being equal to 1, `sub_layer_max_422chroma_constraint_flag[OpTid]` being equal to 1, `sub_layer_max_420chroma_constraint_flag[OpTid]` being equal to 1, `sub_layer_max_monochrome_constraint_flag[OpTid]` being equal to 0, `sub_layer_intra_constraint_flag[OpTid]` being equal to 0, and `sub_layer_one_picture_only_constraint_flag[OpTid]` being equal to 0 and `sub_layer_lower_bit_rate_constraint_flag[OpTid]` being equal to 1.

Conformance of a layer in an output operation point associated with an OLS in a bitstream to the Scalable Main 10 profile is indicated as follows:

- If OpTid of the output operation point is equal to vps_max_sub_layer_minus1, the conformance is indicated by general_profile_idc being equal to 7 or general_profile_compatibility_flag[7] being equal to 1, and general_max_12bit_constraint_flag being equal to 1, general_max_10bit_constraint_flag being equal to 1, general_max_8bit_constraint_flag being equal to 0, general_max_422chroma_constraint_flag being equal to 1, general_max_420chroma_constraint_flag being equal to 1, general_max_monochrome_constraint_flag being equal to 0, general_intra_constraint_flag being equal to 0, general_one_picture_only_constraint_flag being equal to 0 and general_lower_bit_rate_constraint_flag being equal to 1.
- Otherwise (OpTid of the output operation point is less than vps_max_sub_layer_minus1), the conformance is indicated by sub_layer_profile_idc[OpTid] being equal to 7 or sub_layer_profile_compatibility_flag[OpTid][7] being equal to 1, and sub_layer_max_12bit_constraint_flag[OpTid] being equal to 1, sub_layer_max_10bit_constraint_flag[OpTid] being equal to 1, sub_layer_max_8bit_constraint_flag[OpTid] being equal to 0, sub_layer_max_422chroma_constraint_flag[OpTid] being equal to 1, sub_layer_max_420chroma_constraint_flag[OpTid] being equal to 1, sub_layer_max_monochrome_constraint_flag[OpTid] being equal to 0, sub_layer_intra_constraint_flag[OpTid] being equal to 0, and sub_layer_one_picture_only_constraint_flag[OpTid] being equal to 0 and sub_layer_lower_bit_rate_constraint_flag[OpTid] being equal to 1.

H.11.1.2 Scalable format range extensions profiles

The following profiles, collectively referred to as the scalable format range extensions profiles, are specified in this clause:

- The Scalable Monochrome, Scalable Monochrome 12 and Scalable Monochrome 16 profiles
- The Scalable Main 4:4:4 profile

For a layer in an output operation point associated with an OLS in a bitstream, the layer being conforming to the Scalable Monochrome, Scalable Monochrome 12, Scalable Monochrome 16 or Scalable Main 4:4:4 profile, the following applies:

- Let olsIdx be the OLS index of the OLS, the sub-bitstream subBitstream and the base layer sub-bitstream baseBitstream are derived as specified in clause F.11.3.

When vps_base_layer_internal_flag is equal to 1, the base layer sub-bitstream baseBitstream shall obey the following constraints:

- The base layer sub-bitstream baseBitstream shall be indicated to conform to the Main profile, the Main 10 profile or a format range extensions profile.

The sub-bitstream subBitstream shall obey the following constraints:

- All active VPSs shall have vps_num_rep_formats_minus1 in the range of 0 to 15, inclusive.
- All active SPSs for layers in subBitstream shall have separate_colour_plane_flag, cabac_bypass_alignment_enabled_flag, when present, equal to 0 only.
- CtbLog2SizeY derived from all active SPSs for layers in subBitstream shall be in the range of 4 to 6, inclusive.
- The variables NumRefLayerPicsProcessing, NumRefLayerPicsSampleProcessing, and NumRefLayerPicsMotionProcessing shall be less than or equal to 1 for each decoded picture with nuh_layer_id included in layerIdListTarget that was used to derive subBitstream.
- ScalabilityId[j][smIdx] derived according to any active VPS shall be equal to 0 for any smIdx value not equal to 2 or 3 and for any value of j such that layer_id_in_nuh[j] is among layerIdListTarget that was used to derive subBitstream.
- For a layer with nuh_layer_id iNuhLId equal to any value included in layerIdListTarget that was used to derive subBitstream, the value of NumRefLayers[iNuhLId], which specifies the total number of direct and indirect reference layers and is derived as specified in F.7.4.3.1, shall be less than or equal to 4.
- The constraints specified in Table H.3, in which entries marked with "–" indicate that the table entry does not impose a profile-specific constraint on the corresponding syntax element, shall apply for all active SPSs and PPSs for layers in subBitstream.

NOTE – For some syntax elements with table entries marked with "–", a constraint may be imposed indirectly – e.g., by semantics constraints that are imposed elsewhere in this Specification when other specified constraints are fulfilled.

- All active SPSs for layers in subBitstream shall have the same value of chroma_format_idc.
- All active SPSs for layers in subBitstream shall have sps_scc_extension_flag equal to 0 only.
- All active PPSs for layers in subBitstream shall have pps_scc_extension_flag equal to 0 only.

- When an active PPS for any layer in subBitstream has `tiles_enabled_flag` equal to 1, it shall have `entropy_coding_sync_enabled_flag` equal to 0.
- When an active PPS for any layer in subBitstream has `tiles_enabled_flag` equal to 1, `ColumnWidthInLumaSamples[i]` shall be greater than or equal to 256 for all values of `i` in the range of 0 to `num_tile_columns_minus1`, inclusive, and `RowHeightInLumaSamples[j]` shall be greater than or equal to 64 for all values of `j` in the range of 0 to `num_tile_rows_minus1`, inclusive.
- The number of times `read_bits(1)` is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing `coding_tree_unit()` data for any CTU shall be less than or equal to $5 * \text{RawCtuBits} / 3$.
- For any active VPS, `DependencyId[i]` shall be greater than `DependencyId[j]` for any values of `i` and `j` among `layerIdListTarget` that was used to derive subBitstream such that `AuxId[i]` is equal to `AuxId[j]` and `i` is greater than `j`.
- In bitstreams conforming to the Scalable Monochrome, Scalable Monochrome 12, Scalable Monochrome 16 or Scalable Main 4:4:4 profiles, `general_level_idc` and `sub_layer_level_idc[i]` for all values of `i` in active SPSs for all layers shall not be equal to 255 (which indicates level 8.5).
- The tier and level constraints specified for the Scalable Monochrome, Scalable Monochrome 12, Scalable Monochrome 16, or Scalable Main 4:4:4 profiles in clause H.11.2, as applicable, shall be fulfilled.

Table H.3 – Allowed values for syntax elements in the scalable format range extensions profiles

Profile for which constraint is specified	chroma_format_idc	bit_depth_luma_minus8 and bit_depth_chroma_minus8	transform_skip_rotation_enabled_flag, transform_skip_context_enabled_flag, implicit_rdpem_enabled_flag, explicit_rdpem_enabled_flag, intra_smoothing_disabled_flag, persistent_rice_adaptation_enabled_flag, and log2_max_transform_skip_block_size_minus2	extended_precision_processing_flag	chroma_qp_offset_list_enabled_flag
Scalable Monochrome	0	0	0	0	0
Scalable Monochrome 12	0	0..4	0	0	0
Scalable Monochrome 16	0	–	–	–	0
Scalable Main 4:4:4	–	0	–	0	–

In the remainder of this clause and clause H.11.2.1, all syntax elements in the `profile_tier_level()` syntax structure refer to those in the `profile_tier_level()` syntax structure associated with the layer.

Conformance of a layer in an output operation point associated with an OLS in a bitstream for the scalable format range extensions profiles is indicated as follows:

- If `OpTid` of the output operation point is equal to `vps_max_sub_layer_minus1`, the conformance is indicated by `general_profile_idc` being equal to 10 or `general_profile_compatibility_flag[10]` being equal to 1, with the additional indications specified in Table H.4 for the general constraint flags.
- Otherwise (`OpTid` of the output operation point is less than `vps_max_sub_layer_minus1`), the conformance is indicated by `sub_layer_profile_idc[OpTid]` being equal to 10 or `sub_layer_profile_compatibility_flag[OpTid][10]` being equal to 1, with the additional indications specified in Table H.4 for the flags associated with the index `OpTid`.

All other combinations of `general_max_14bit_constraint_flag`, `general_max_12bit_constraint_flag`, `general_max_10bit_constraint_flag`, `general_max_8bit_constraint_flag`, `general_max_422chroma_constraint_flag`, `general_max_420chroma_constraint_flag`, `general_max_monochrome_constraint_flag`, `general_intra_constraint_flag`, `general_one_picture_only_constraint_flag`, and `general_lower_bit_rate_constraint_flag` with `general_profile_idc` equal to 10 or

general_profile_compatibility_flag[10] equal to 1 are reserved for future use by ITU-T | ISO/IEC. All other combinations of sub_layer_max_14bit_constraint_flag[OpTid], sub_layer_max_12bit_constraint_flag[OpTid], sub_layer_max_10bit_constraint_flag[OpTid], sub_layer_max_8bit_constraint_flag[OpTid], sub_layer_max_422chroma_constraint_flag[OpTid], sub_layer_max_420chroma_constraint_flag[OpTid], sub_layer_max_monochrome_constraint_flag[OpTid], sub_layer_intra_constraint_flag[OpTid], sub_layer_one_picture_only_constraint_flag[OpTid], and sub_layer_lower_bit_rate_constraint_flag[OpTid] with sub_layer_profile_idc[OpTid] equal to 10 or sub_layer_profile_compatibility_flag[OpTid][10] equal to 1 are reserved for future use by ITU-T | ISO/IEC. Such combinations shall not be present in bitstreams conforming to this Specification. However, decoders conforming to the scalable format range extensions profiles shall allow other combinations as specified below in this clause to occur in the bitstream.

Table H.4 – Bitstream indications for conformance to scalable range extensions profiles

Profile for which the bitstream indicates conformance	general_max_14bit_constraint_flag or sub_layer_max_14bit_constraint_flag[OpTid]	general_max_12bit_constraint_flag or sub_layer_max_12bit_constraint_flag[OpTid]	general_max_10bit_constraint_flag or sub_layer_max_10bit_constraint_flag[OpTid]	general_max_8bit_constraint_flag or sub_layer_max_8bit_constraint_flag[OpTid]	general_max_422chroma_constraint_flag or sub_layer_max_422chroma_constraint_flag[OpTid]	general_max_420chroma_constraint_flag or sub_layer_max_420chroma_constraint_flag[OpTid]	general_max_monochrome_constraint_flag or sub_layer_max_monochrome_constraint_flag[OpTid]	general_intra_constraint_flag or sub_layer_intra_constraint_flag[OpTid]	general_one_picture_only_constraint_flag or sub_layer_one_picture_only_constraint_flag[OpTid]	general_lower_bit_rate_constraint_flag or sub_layer_lower_bit_rate_constraint_flag[OpTid]
Scalable Monochrome	1	1	1	1	1	1	1	0	0	1
Scalable Monochrome 12	1	1	0	0	1	1	1	0	0	1
Scalable Monochrome 16	0	0	0	0	1	1	1	0	0	1
Scalable Main 4:4:4	1	1	1	1	0	0	0	0	0	1

H.11.2 Tiers and levels

H.11.2.1 General tier and level limits

For purposes of comparison of tier capabilities, the tier with general_tier_flag or sub_layer_tier_flag[i] equal to 0 is considered to be a lower tier than the tier with general_tier_flag or sub_layer_tier_flag[i] equal to 1.

For purposes of comparison of level capabilities, a particular level of a specific tier is considered to be a lower level than some other level of the same tier when the value of the general_level_idc or sub_layer_level_idc[i] of the particular level is less than that of the other level.

The following is specified for expressing the constraints in this clause and clause H.11.2.2:

- For the Scalable Main profile, the value of each of the variables CpbVclFactor, CpbNalFactor, FormatCapabilityFactor and MinCrScaleFactor is the same as that specified in Table A.10 for the Main profile. For the Scalable Main 10 profile, the value of each of these variables is the same as that specified in Table A.10 for the Main 10 profile.
- For the Scalable Monochrome profile, the value of each of the variables CpbVclFactor, CpbNalFactor, FormatCapabilityFactor, and MinCrScaleFactor is the same as that specified in Table A.10 for the Monochrome profile.
- For the Scalable Monochrome 12 profile, the value of each of the variables CpbVclFactor, CpbNalFactor, FormatCapabilityFactor, and MinCrScaleFactor is the same as that specified in Table A.10 for the Monochrome 12 profile.

- For the Scalable Monochrome 16 profile, the value of each of the variables CpbVclFactor, CpbNalFactor, FormatCapabilityFactor, and MinCrScaleFactor is the same as that specified in Table A.10 for the Monochrome 16 profile.
- For the Scalable Main 4:4:4 profile, the value of each of the variables CpbVclFactor, CpbNalFactor, FormatCapabilityFactor, and MinCrScaleFactor is the same as that specified in Table A.10 for the Main 4:4:4 profile.
- Let access unit n be the n -th access unit in decoding order, with the first access unit being access unit 0 (i.e., the 0-th access unit).
- Let the variable fR be set equal to $1 \div 300$.
- Let the variable $olsIdx$ be the index of the OLS.
- For each layer with nuh_layer_id equal to $currLayerId$, let the variable $layerSizeInSamplesY$ be derived as follows:

$$layerSizeInSamplesY = pic_width_vps_in_luma_samples * pic_height_vps_in_luma_samples \quad (H-121)$$

where $pic_width_vps_in_luma_samples$ and $pic_height_vps_in_luma_samples$ are found in the $vps_rep_format_idx[LayerIdxInVps[currLayerId]]$ -th $rep_format()$ syntax structure in the VPS.

Each layer with nuh_layer_id equal to $currLayerId$ conforming to a profile at a specified tier and level shall obey the following constraints for each bitstream conformance test as specified in clause F.13, where "access unit" is used to denote the picture unit in the layer and the CPB is understood to be the BPB:

- a) The value of $layerSizeInSamplesY$ shall be less than or equal to $MaxLumaPs$, where $MaxLumaPs$ is specified in Table A.8 for the tier and level of the layer.
- b) The value of $pic_width_vps_in_luma_samples$ of the $vps_rep_format_idx[LayerIdxInVps[currLayerId]]$ -th $rep_format()$ syntax structure in the VPS shall be less than or equal to $Sqrt(MaxLumaPs * 8)$.
- c) The value of $pic_height_vps_in_luma_samples$ of the $vps_rep_format_idx[LayerIdxInVps[currLayerId]]$ -th $rep_format()$ syntax structure in the VPS shall be less than or equal to $Sqrt(MaxLumaPs * 8)$.
- d) The value of $max_vps_dec_pic_buffering_minus1[olsIdx][LayerIdxInVps[currLayerId]][HighestTid]$ shall be less than or equal to $MaxDpbSize$ as derived by Equation A-2, with $PicSizeInSamplesY$ being replaced with $layerSizeInSamplesY$, for the tier and level of the layer.
- e) For level 5 and higher levels, the value of $CtbSizeY$ for the layer shall be equal to 32 or 64.
- f) The value of $NumPicTotalCurr$ for each picture in the layer shall be less than or equal to 8.
- g) When decoding each coded picture in the layer, the value of $num_tile_columns_minus1$ shall be less than $MaxTileCols$ and $num_tile_rows_minus1$ shall be less than $MaxTileRows$, where $MaxTileCols$ and $MaxTileRows$ are specified in Table A.8 for the tier and level of the layer.
- h) For the VCL HRD parameters of the layer, $CpbSize[i]$ shall be less than or equal to $CpbVclFactor * MaxCPB$ for at least one of the delivery schedules identified by $bsp_sched_idx[olsIdx][0][HighestTid][combIdx][LayerIdxInVps[currLayerId]]$ for $combIdx$ ranging from 0 to $num_bsp_schedules_minus1[olsIdx][0][HighestTid]$, inclusive, where $CpbSize[i]$ is specified in clause F.13.1 and $MaxCPB$ is specified in Table A.8 for the tier and level of the layer in units of $CpbVclFactor$ bits.
- i) For the NAL HRD parameters of the layer, $CpbSize[i]$ shall be less than or equal to $CpbNalFactor * MaxCPB$ for at least one of the delivery schedules identified by $bsp_sched_idx[olsIdx][0][HighestTid][combIdx][LayerIdxInVps[currLayerId]]$ for $combIdx$ ranging from 0 to $num_bsp_schedules_minus1[olsIdx][0][HighestTid]$, inclusive, where $CpbSize[i]$ is specified in clause F.13.1 and $MaxCPB$ is specified in Table A.8 for the tier and level of the layer in units of $CpbNalFactor$ bits.

Table A.8 specifies the limits for each level of each tier for levels other than level 8.5.

A tier and level to which a layer in an output operation point associated with an OLS in a bitstream conforms are indicated by the syntax elements $general_tier_flag$ and $general_level_idc$ if $OpTid$ of the output layer set is equal to

vps_max_sub_layer_minus1, and by the syntax elements sub_layer_tier_flag[OpTid] and sub_layer_level_idc[OpTid] otherwise, as follows:

- If the specified level is not level 8.5, general_tier_flag or sub_layer_tier_flag[OpTid] equal to 0 indicates conformance to the Main tier, and general_tier_flag or sub_layer_tier_flag[OpTid] equal to 1 indicates conformance to the High tier, according to the tier constraints specified in Table A.8, and general_tier_flag and sub_layer_tier_flag[OpTid] shall be equal to 0 for levels below level 4 (corresponding to the entries in Table A.8 marked with "-"). Otherwise (the specified level is level 8.5), it is a requirement of bitstream conformance that general_tier_flag and sub_layer_tier_flag[OpTid] shall be equal to 1 and the value 0 for general_tier_flag and sub_layer_tier_flag[OpTid] is reserved for future use by ITU-T | ISO/IEC and decoders shall ignore the value of general_tier_flag and sub_layer_tier_flag[OpTid].
- general_level_idc and sub_layer_level_idc[OpTid] shall be set equal to a value of 30 times the level number specified in Table A.8.

H.11.2.2 Profile-specific tier and level limits for the Scalable Main, Scalable Main 10 and scalable format range extensions profiles

The following is specified for expressing the constraints in this clause:

- The variable HbrFactor is set equal to 1.
- The variable BrVclFactor is set equal to CpbVclFactor * HbrFactor.
- The variable BrNalFactor is set equal to CpbNalFactor * HbrFactor.
- The variable MinCr is set equal to MinCrBase * MinCrScaleFactor ÷ HbrFactor, where MinCrBase is specified in Table A.9.

Each layer conforming to the Scalable Main or Scalable Main 10 profiles or a scalable format range extensions profile at a specified tier and level shall obey the following constraints for each conformance test as specified in clause F.13, where "access unit" is used to denote the picture unit in the layer and the CPB is understood to be the BPB:

- a) The nominal removal time of access unit n (with n greater than 0) from the CPB, as specified in clause F.13.2.3, shall satisfy the constraint that $AuNominalRemovalTime[n] - AuCpbRemovalTime[n - 1]$ is greater than or equal to $\text{Max}(\text{layerSizeInSamplesY} \div \text{MaxLumaSr}, fR)$, where $\text{layerSizeInSamplesY}$ is the value of $\text{layerSizeInSamplesY}$ for access unit $n - 1$ and MaxLumaSr is the value specified in Table A.9 that applies to access unit $n - 1$ for the tier and level of the layer.
- b) The difference between consecutive output times of pictures in different access units, as specified in clause F.13.3.3 shall satisfy the constraint that $\text{DpbOutputInterval}[n]$ is greater than or equal to $\text{Max}(\text{layerSizeInSamplesY} \div \text{MaxLumaSr}, fR)$, where $\text{layerSizeInSamplesY}$ is the value of $\text{layerSizeInSamplesY}$ of access unit n and MaxLumaSr is the value specified in Table A.9 for access unit n for the tier and level of the layer, provided that access unit n is an access unit that has a picture that is output and is not the last of such access units.
- c) The removal time of access unit 0 shall satisfy the constraint that the number of coded slice segments in access unit 0 is less than or equal to $\text{Min}(\text{Max}(1, \text{MaxSliceSegmentsPerPicture} * \text{MaxLumaSr} / \text{MaxLumaPs} * (\text{AuCpbRemovalTime}[0] - \text{AuNominalRemovalTime}[0]) + \text{MaxSliceSegmentsPerPicture} * \text{layerSizeInSamplesY} / \text{MaxLumaPs}), \text{MaxSliceSegmentsPerPicture})$, for the value of $\text{layerSizeInSamplesY}$ of access unit 0, where $\text{MaxSliceSegmentsPerPicture}$, MaxLumaPs and MaxLumaSr are the values specified in Table A.8 and Table A.9 for the tier and level of the layer.
- d) The difference between consecutive CPB removal times of access units n and $n - 1$ (with n greater than 0) shall satisfy the constraint that the number of slice segments in access unit n is less than or equal to $\text{Min}(\text{Max}(1, \text{MaxSliceSegmentsPerPicture} * \text{MaxLumaSr} / \text{MaxLumaPs} * (\text{AuCpbRemovalTime}[n] - \text{AuCpbRemovalTime}[n - 1])), \text{MaxSliceSegmentsPerPicture})$, where $\text{MaxSliceSegmentsPerPicture}$, MaxLumaPs and MaxLumaSr are the values specified in Table A.8 and Table A.9 that apply to access unit n for the tier and level of the layer.
- e) For the VCL HRD parameters for the layer, $\text{BitRate}[i]$ shall be less than or equal to $\text{BrVclFactor} * \text{MaxBR}$ for at least one of the delivery schedules identified by $\text{bsp_sched_idx}[\text{olsIdx}][0][\text{HighestTid}][\text{combIdx}][\text{LayerIdxInVps}[\text{currLayerId}]]$ for combIdx ranging from 0 to $\text{num_bsp_schedules_minus1}[\text{olsIdx}][0][\text{HighestTid}]$, inclusive, where $\text{BitRate}[i]$ is specified in clause F.13.1 and MaxBR is specified in Table A.9 in units of BrVclFactor bits/s for the tier and level of the layer.
- f) For the NAL HRD parameters for the layer, $\text{BitRate}[i]$ shall be less than or equal to $\text{BrNalFactor} * \text{MaxBR}$ for at least one of the delivery schedules identified by $\text{bsp_sched_idx}[\text{olsIdx}][0][\text{HighestTid}][\text{combIdx}][\text{LayerIdxInVps}[\text{currLayerId}]]$ for combIdx ranging from 0 to $\text{num_bsp_schedules_minus1}[\text{olsIdx}][0]$

[HighestTid], inclusive, where BitRate[i] is specified in clause F.13.1 and MaxBR is specified in Table A.9 in units of BrNalFactor bits/s for the tier and level of the layer.

- g) The sum of the NumBytesInNalUnit variables for access unit 0 shall be less than or equal to $\text{FormatCapabilityFactor} * (\text{Max}(\text{layerSizeInSamplesY}, \text{fR} * \text{MaxLumaSr}) + \text{MaxLumaSr} * (\text{AuCpbRemovalTime}[0] - \text{AuNominalRemovalTime}[0])) \div \text{MinCr}$ for the value of layerSizeInSamplesY of access unit 0, where MaxLumaSr is specified in Table A.9, and both MaxLumaSr and FormatCapabilityFactor are the values that apply to access unit 0 for the tier and level of the layer.
- h) The sum of the NumBytesInNalUnit variables for access unit n (with n greater than 0) shall be less than or equal to $\text{FormatCapabilityFactor} * \text{MaxLumaSr} * (\text{AuCpbRemovalTime}[n] - \text{AuCpbRemovalTime}[n-1]) \div \text{MinCr}$, where MaxLumaSr is specified in Table A.9, and both MaxLumaSr and FormatCapabilityFactor are the values that apply to access unit n for the tier and level of the layer.
- i) The removal time of access unit 0 shall satisfy the constraint that the number of tiles in coded pictures in access unit 0 is less than or equal to $\text{Min}(\text{Max}(1, \text{MaxTileCols} * \text{MaxTileRows} * 120 * (\text{AuCpbRemovalTime}[0] - \text{AuNominalRemovalTime}[0]) + \text{MaxTileCols} * \text{MaxTileRows} * \text{PicSizeInSamplesY} / \text{MaxLumaPs}), \text{MaxTileCols} * \text{MaxTileRows})$, for the value of layerSizeInSamplesY of access unit 0, where MaxTileCols and MaxTileRows are the values specified in Table A.8 that apply to access unit 0 for the tier and level of the layer.
- j) The difference between consecutive CPB removal times of access units n and n – 1 (with n greater than 0) shall satisfy the constraint that the number of tiles in coded pictures in access unit n is less than or equal to $\text{Min}(\text{Max}(1, \text{MaxTileCols} * \text{MaxTileRows} * 120 * (\text{AuCpbRemovalTime}[n] - \text{AuCpbRemovalTime}[n-1])), \text{MaxTileCols} * \text{MaxTileRows})$, where MaxTileCols and MaxTileRows are the values specified in Table A.8 that apply to access unit n for the tier and level of the layer.

H.11.3 Decoder capabilities

When a decoder conforms to any profile specified in Annex H, it shall also have the INBLD capability specified in clause F.11.1.

Clause F.11.2 specifies requirements for a decoder conforming to any profile specified in this annex.

H.12 Byte stream format

The specifications in clause F.12 apply.

H.13 Hypothetical reference decoder

The specifications in clause F.13 and its subclauses apply.

H.14 Supplemental enhancement information

The specifications in Annex D and clause F.14 and their subclauses apply.

H.15 Video usability information

The specifications in clause F.15 and its subclauses apply.

Annex I

3D high efficiency video coding

(This annex forms an integral part of this Recommendation | International Standard.)

I.1 Scope

This annex specifies syntax, semantics, and decoding processes for 3D high efficiency video coding that use the syntax, semantics, and decoding processes specified in clauses 2-10 and Annexes A-G. This annex also specifies profiles, tiers, and levels for 3D high efficiency video coding.

I.2 Normative references

The list of normative references in clause G.2 apply.

I.3 Definitions

For the purpose of this annex, the following definitions apply in addition to the definitions in clause G.3. These definitions are either not present in clause G.3 or replace definitions in clause G.3.

- I.3.1 depth intra contour prediction:** A *prediction* of a *partition pattern* for a *prediction block* in a *picture* of a *depth layer* derived from samples of a *picture* included in the same *access unit* and in the *texture layer* of the same *view*.
- I.3.2 depth layer:** A *layer* with a *nuh_layer_id* value equal to *i*, such that *DepthLayerFlag[i]* is equal to 1 and *DependencyId[i]* and *AuxId[i]* are equal to 0.
- I.3.3 depth look-up table:** A list containing *depth values*.
- I.3.4 depth value:** A sample value of a *decoded picture* of a *depth layer*.
- I.3.5 disparity vector:** A *motion vector* used for inter-view prediction.
- I.3.6 inter-component prediction:** An *inter-layer prediction* where the *reference pictures* are associated with a *DepthFlag* value different from the *DepthFlag* value of the current *picture*.
- I.3.7 inter-view prediction:** An *inter-layer prediction* where the *reference pictures* are associated with *reference view order index* values different from the *ViewIdx* value of the current *picture*.
- I.3.8 intra prediction:** A *prediction* derived from only data elements (e.g., sample values) of the same *decoded slice* and additionally *may* be using *depth intra contour prediction*.
- I.3.9 partition pattern:** An MxM (M-column by M-row) array of *flags* defining two *sub-block partitions* of an MxM *prediction block*.
- I.3.10 prediction block:** A rectangular MxN *block* of samples on which either the same *prediction* or *partitioning* in *sub-block partitions* is applied.
- I.3.11 reference view order index:** A *ViewIdx* value associated with a *reference picture* used for *inter-view prediction*.
- I.3.12 sub-block partition:** A subset of samples of a *prediction block* on which the same *prediction* is applied.
- I.3.13 texture layer:** A *layer* with a *nuh_layer_id* value equal to *i*, such that *DepthLayerFlag[i]*, *DependencyId[i]*, and *AuxId[i]* are equal to 0.

I.4 Abbreviations

The specifications in clause G.4 apply.

I.5 Conventions

The specifications in clause G.5 apply.

I.6 Bitstream and picture formats, partitionings, scanning processes, and neighbouring relationships

I.6.1 Bitstream formats

The specifications in clause 6.1 apply.

I.6.2 Source, decoded, and output picture formats

The specifications in clause 6.2 apply.

I.6.3 Partitioning of pictures, slices, slice segments, tiles, CTUs, and CTBs

The specifications in clause 6.3 and its subclauses apply.

I.6.4 Availability processes

The specifications in clause 6.4 apply.

I.6.5 Scanning processes

The specifications in clause 6.5 and its subclauses apply.

I.6.6 Derivation process for a wedgelet partition pattern table

NOTE – Tables and values resulting from this process are independent of any information contained in the bitstream.

The list `WedgePatternTable[log2BlkSize]` of partition patterns of size $(1 \ll \log2BlkSize) \times (1 \ll \log2BlkSize)$ and the variable `NumWedgePattern[log2BlkSize]` specifying the number of partition patterns in list `WedgePatternTable[log2BlkSize]` are derived as follows:

- For `log2BlkSize` in the range of 2 to 4, inclusive, the following applies:
 - `NumWedgePattern[log2BlkSize]` is set equal to 0.
 - The variable `resShift` is set equal to $(\log2BlkSize == 4) ? 0 : 1$.
 - The variable `wBlkSize` is set equal to $(1 \ll (\log2BlkSize + resShift))$.
 - For `wedgeOri` in the range of 0 to 5, inclusive, the following applies:
 - The variable `posEnd` is set equal to `NumWedgePattern[log2BlkSize]`.
 - If `wedgeOri` is equal to 0 or 4, the following applies:
 - The variables `sizeScaleS` and `sizeScaleE` are derived as follows:
$$sizeScaleS = (\log2BlkSize > 3) ? 2 : 1 \quad (I-1)$$
$$sizeScaleE = (\text{wedgeOri} < 4 \ \&\& \ \log2BlkSize > 3) ? 2 : 1 \quad (I-2)$$
 - For `m` in the range of 0 to $(wBlkSize / sizeScaleS - 1)$, inclusive, the following applies:
 - For `n` in the range of 0 to $(wBlkSize / sizeScaleE - 1)$, inclusive, the following applies:
 - The wedgelet partition pattern generation process as specified in clause I.6.6.1 is invoked with `patternSize` equal to $(1 \ll \log2BlkSize)$, the variable `resShift`, the variable `wedgeOri`, the variable (xS, yS) equal to $(m * sizeScaleS, 0)$, the variable (xE, yE) equal to $(\text{wedgeOri} == 0) ? (0, n * sizeScaleE) : (n * sizeScaleE, wBlkSize - 1)$ as inputs, and the output is the partition pattern `curWedgePattern`.
 - The wedgelet partition pattern table insertion process as specified in clause I.6.6.2 is invoked with the variable `log2BlkSize` and the partition pattern `curWedgePattern` as inputs.
 - Otherwise (`wedgeOri` is equal to 1, 2, 3, or 5), the following applies:
 - For `curPos` in the range of `posStart` to `posEnd - 1`, inclusive, the following applies:
 - The partition pattern `curWedgePattern[x][y]` is derived as follows:
$$\begin{aligned} &\text{for}(y = 0; y < (1 \ll \log2BlkSize); y++) \\ &\quad \text{for}(x = 0; x < (1 \ll \log2BlkSize); x++) \\ &\quad \quad \text{curWedgePattern}[x][y] = 1 - \\ &\quad \quad \quad \text{WedgePatternTable}[\log2BlkSize][\text{curPos}][y][(1 \ll \log2BlkSize) - 1 - x] \end{aligned} \quad (I-3)$$
 - The variable `posStart` is set equal to `posEnd`.
 - `NumWedgePattern[5]` is set equal to `NumWedgePattern[4]`.
 - For `k = 0..NumWedgePattern[5] - 1`, the following applies:
 - For `x, y = 0..(1 << 5) - 1`, the following applies:
$$\text{WedgePatternTable}[5][k][x][y] = \text{WedgePatternTable}[4][k][x >> 1][y >> 1] \quad (I-4)$$

I.6.6.1 Wedgelet partition pattern generation process

Inputs to this process are:

- a variable `patternSize` specifying the partition pattern size,
- a variable `resShift` specifying the precision of the partition pattern start and end locations relative to `patternSize`,
- a variable `wedgeOri` specifying the orientation of the partition pattern,
- a location (`xS`, `yS`) specifying the boundary start of a sub-block partition,
- a location (`xE`, `yE`) specifying the boundary end of a sub-block partition.

Output of this process is the partition pattern `wedgePattern[x][y]` of size (`patternSize`)x(`patternSize`).

The values of the partition pattern `wedgePattern[x][y]` are derived as specified by the following ordered steps:

1. For `x, y = 0..patternSize - 1`, `wedgePattern[x][y]` is set equal to 0.
2. The samples of the partition pattern `wedgePattern` that form a line between (`xS`, `yS`) and (`xE`, `yE`) are set equal to 1 as follows:

```
( x0, y0 ) = ( xS, yS )
( x1, y1 ) = ( xE, yE )
if( abs( yE - yS ) > abs( xE - xS ) ) {
    ( x0, y0 ) = Swap( x0, y0 )
    ( x1, y1 ) = Swap( x1, y1 )
}
if( x0 > x1 ) {
    ( x0, x1 ) = Swap( x0, x1 )
    ( y0, y1 ) = Swap( y0, y1 )
}
sumErr = 0
posY = y0
for( posX = x0; posX <= x1; posX++ ) {
    if( abs( yE - yS ) > abs( xE - xS ) )
        wedgePattern[ posY >> resShift ][ posX >> resShift ] = 1
    else
        wedgePattern[ posX >> resShift ][ posY >> resShift ] = 1
    sumErr += ( abs( y1 - y0 ) << 1 )
    if( sumErr >= ( x1 - x0 ) ) {
        posY += ( y0 < y1 ) ? 1 : -1
        sumErr -= ( x1 - x0 ) << 1
    }
}
```

(I-5)

3. The samples of `wedgePattern` are modified as follows:

```
for( y = 0; y <= ( yE >> resShift ); y++ )
    for( x = 0; ( x <= patternSize - 1 ) && ( wedgePattern[ x ][ y ] == 0 ); x++ )
        wedgePattern[ x ][ y ] = 1
```

(I-6)

I.6.6.2 Wedgelet partition pattern table insertion process

Inputs to this process are:

- a variable `log2BlkSize` specifying the partition pattern size,
- a partition pattern `wedgePattern[x][y]`, with `x, y = 0..(1 << log2BlkSize) - 1`.

The variable `validPatternFlag` is set equal to 0 and the following applies:

1. For `x, y = 0..(1 << log2BlkSize) - 1`, the following applies:
 - When `wedgePattern[x][y]` is not equal to `wedgePattern[0][0]`, `validPatternFlag` is set equal to 1.
2. For `k = 0..NumWedgePattern[log2BlkSize] - 1`, the following applies:
 - The variable `patIdenticalFlag` is set equal to 1.
 - For `x, y = 0..(1 << log2BlkSize) - 1`, the following applies:

- When `wedgePattern[x][y]` is not equal to `WedgePatternTable[log2BlkSize][k][x][y]`, `patIdenticalFlag` is set equal to 0.
 - When `patIdenticalFlag` is equal to 1, `validPatternFlag` is set equal to 0.
3. For `k = 0..NumWedgePattern[log2BlkSize] – 1`, the following applies:
- The variable `patInvIdenticalFlag` is set equal to 1.
 - For `x, y = 0..(1 << log2BlkSize) – 1`, the following applies:
 - When `wedgePattern[x][y]` is equal to `WedgePatternTable[log2BlkSize][k][x][y]`, `patInvIdenticalFlag` is set equal to 0.
 - When `patInvIdenticalFlag` is equal to 1, `validPatternFlag` is set equal to 0.

When `validPatternFlag` is equal to 1, the following applies:

- The pattern `WedgePatternTable[log2BlkSize][NumWedgePattern[log2BlkSize]]` is set equal to `wedgePattern`.
- The value of `NumWedgePattern[log2BlkSize]` is incremented by one.

I.7 Syntax and semantics

I.7.1 Method of specifying syntax in tabular form

The specifications in clause F.7.1 apply.

I.7.2 Specification of syntax functions, categories, and descriptors

The specifications in clause F.7.2 apply.

I.7.3 Syntax in tabular form

I.7.3.1 NAL unit syntax

The specifications in clause F.7.3.1 and all its subclauses apply.

I.7.3.2 Raw byte sequence payloads and RBSP trailing bits syntax

I.7.3.2.1 Video parameter set RBSP

<code>video_parameter_set_rbsp() {</code>	Descriptor
<code> vps_video_parameter_set_id</code>	<code>u(4)</code>
<code> vps_base_layer_internal_flag</code>	<code>u(1)</code>
<code> vps_base_layer_available_flag</code>	<code>u(1)</code>
<code> vps_max_layers_minus1</code>	<code>u(6)</code>
<code> vps_max_sub_layers_minus1</code>	<code>u(3)</code>
<code> vps_temporal_id_nesting_flag</code>	<code>u(1)</code>
<code> vps_reserved_0xffff_16bits</code>	<code>u(16)</code>
<code> profile_tier_level(1, vps_max_sub_layers_minus1)</code>	
<code> vps_sub_layer_ordering_info_present_flag</code>	<code>u(1)</code>
<code> for(i = (vps_sub_layer_ordering_info_present_flag ? 0 : vps_max_sub_layers_minus1);</code> <code> i <= vps_max_sub_layers_minus1; i++) {</code>	
<code> vps_max_dec_pic_buffering_minus1[i]</code>	<code>ue(v)</code>
<code> vps_max_num_reorder_pics[i]</code>	<code>ue(v)</code>
<code> vps_max_latency_increase_plus1[i]</code>	<code>ue(v)</code>
<code> }</code>	
<code> vps_max_layer_id</code>	<code>u(6)</code>
<code> vps_num_layer_sets_minus1</code>	<code>ue(v)</code>
<code> for(i = 1; i <= vps_num_layer_sets_minus1; i++)</code>	
<code> for(j = 0; j <= vps_max_layer_id; j++)</code>	
<code> layer_id_included_flag[i][j]</code>	<code>u(1)</code>

vps_timing_info_present_flag	u(1)
if(vps_timing_info_present_flag) {	
vps_num_units_in_tick	u(32)
vps_time_scale	u(32)
vps_poc_proportional_to_timing_flag	u(1)
if(vps_poc_proportional_to_timing_flag)	
vps_num_ticks_poc_diff_one_minus1	ue(v)
vps_num_hrd_parameters	ue(v)
for(i = 0; i < vps_num_hrd_parameters; i++) {	
hrd_layer_set_idx[i]	ue(v)
if(i > 0)	
cprms_present_flag[i]	u(1)
hrd_parameters(cprms_present_flag[i], vps_max_sub_layers_minus1)	
}	
}	
vps_extension_flag	u(1)
if(vps_extension_flag) {	
while(!byte_aligned())	
vps_extension_alignment_bit_equal_to_one	u(1)
vps_extension()	
vps_extension2_flag	u(1)
if(vps_extension2_flag) {	
vps_3d_extension_flag	u(1)
if(vps_3d_extension_flag) {	
while(!byte_aligned())	
vps_3d_extension_alignment_bit_equal_to_one	u(1)
vps_3d_extension()	
}	
vps_extension3_flag	u(1)
if(vps_extension3_flag)	
while(more_rbsp_data())	
vps_extension_data_flag	u(1)
}	
}	
rbp_trailing_bits()	
}	

I.7.3.2.1.1 Video parameter set extension syntax

The specifications in clause F.7.3.2.1.1 apply.

I.7.3.2.1.2 Representation format syntax

The specifications in clause F.7.3.2.1.2 apply.

I.7.3.2.1.3 DPB size syntax

The specifications in clause F.7.3.2.1.3 apply.

I.7.3.2.1.4 VPS VUI syntax

The specifications in clause F.7.3.2.1.4 apply.

I.7.3.2.1.5 Video signal info syntax

The specifications in clause F.7.3.2.1.5 apply.

I.7.3.2.1.6 VPS VUI bitstream partition HRD parameters syntax

The specifications in clause F.7.3.2.1.6 apply.

I.7.3.2.1.7 Video parameter set 3D extension syntax

vps_3d_extension() {	Descriptor
cp_precision	ue(v)
for(n = 1; n < NumViews; n++) {	
i = ViewOldxList[n]	
num_cp [i]	u(6)
if(num_cp[i] > 0) {	
cp_in_slice_segment_header_flag [i]	u(1)
for(m = 0; m < num_cp[i]; m++) {	
cp_ref_voi [i][m]	ue(v)
if(!cp_in_slice_segment_header_flag[i]) {	
j = cp_ref_voi[i][m]	
vps_cp_scale [i][j]	se(v)
vps_cp_off [i][j]	se(v)
vps_cp_inv_scale_plus_scale [i][j]	se(v)
vps_cp_inv_off_plus_off [i][j]	se(v)
}	
}	
}	
}	

I.7.3.2.2 Sequence parameter set RBSP syntax

I.7.3.2.2.1 General sequence parameter set RBSP syntax

The specifications in clause F.7.3.2.2.1 apply.

I.7.3.2.2.2 Sequence parameter set range extension syntax

The specifications in clause F.7.3.2.2.2 apply.

I.7.3.2.2.3 Sequence parameter set screen content coding extension syntax

The specifications in clause F.7.3.2.2.3 apply.

I.7.3.2.2.4 Sequence parameter set multilayer extension syntax

The specifications in clause F.7.3.2.2.4 apply.

I.7.3.2.2.5 Sequence parameter set 3D extension syntax

sps_3d_extension() {	Descriptor
for(d = 0; d <= 1; d++) {	
iv_di_mc_enabled_flag [d]	u(1)
iv_mv_scal_enabled_flag [d]	u(1)
if(d == 0) {	
log2_ivmc_sub_pb_size_minus3 [d]	ue(v)

iv_res_pred_enabled_flag[d]	u(1)
depth_ref_enabled_flag[d]	u(1)
vsp_mc_enabled_flag[d]	u(1)
dbbp_enabled_flag[d]	u(1)
} else {	
tex_mc_enabled_flag[d]	u(1)
log2_texmc_sub_pb_size_minus3[d]	ue(v)
intra_contour_enabled_flag[d]	u(1)
intra_dc_only_wedge_enabled_flag[d]	u(1)
cqt_cu_part_pred_enabled_flag[d]	u(1)
inter_dc_only_enabled_flag[d]	u(1)
skip_intra_enabled_flag[d]	u(1)
}	
}	
}	

I.7.3.2.3 Picture parameter set RBSP syntax

I.7.3.2.3.1 General picture parameter set RBSP syntax

The specifications in clause F.7.3.2.3.1 apply.

I.7.3.2.3.2 Picture parameter set range extension syntax

The specifications in clause F.7.3.2.3.2 apply.

I.7.3.2.3.3 Picture parameter set screen content coding extension syntax

The specifications in clause F.7.3.2.3.3 apply.

I.7.3.2.3.4 Picture parameter set multilayer extension syntax

The specifications in clause F.7.3.2.3.4 apply.

I.7.3.2.3.5 General colour mapping table syntax

The specifications in clause F.7.3.2.3.5 apply.

I.7.3.2.3.6 Colour mapping octants syntax

The specifications in clause F.7.3.2.3.6 apply.

I.7.3.2.3.7 Picture parameter set 3D extension syntax

pps_3d_extension() {	Descriptor
dlts_present_flag	u(1)
if(dlts_present_flag) {	
pps_depth_layers_minus1	u(6)
pps_bit_depth_for_depth_layers_minus8	u(4)
for(i = 0; i <= pps_depth_layers_minus1; i++) {	
dlt_flag[i]	u(1)
if(dlt_flag[i]) {	
dlt_pred_flag[i]	u(1)
if(!dlt_pred_flag[i])	
dlt_val_flags_present_flag[i]	u(1)
if(dlt_val_flags_present_flag[i])	
for(j = 0; j <= depthMaxValue; j++)	

dlt_value_flag[i][j]	u(1)
else	
delta_dlt(i)	
}	
}	
}	
}	

I.7.3.2.3.8 Delta depth look-up table syntax

delta_dlt(i) {	Descriptor
num_val_delta_dlt	u(v)
if(num_val_delta_dlt > 0) {	
if(num_val_delta_dlt > 1)	
max_diff	u(v)
if(num_val_delta_dlt > 2 && max_diff > 0)	
min_diff_minus1	u(v)
delta_dlt_val0	u(v)
if(max_diff > (min_diff_minus1 + 1))	
for(k = 1; k < num_val_delta_dlt; k++)	
delta_val_diff_minus_min[k]	u(v)
}	
}	

I.7.3.2.4 Supplemental enhancement information RBSP syntax

The specifications in clause F.7.3.2.4 apply.

I.7.3.2.5 Access unit delimiter RBSP syntax

The specifications in clause F.7.3.2.5 apply.

I.7.3.2.6 End of sequence RBSP syntax

The specifications in clause F.7.3.2.6 apply.

I.7.3.2.7 End of bitstream RBSP syntax

The specifications in clause F.7.3.2.7 apply.

I.7.3.2.8 Filler data RBSP syntax

The specifications in clause F.7.3.2.8 apply.

I.7.3.2.9 Slice segment layer RBSP syntax

The specifications in clause F.7.3.2.9 apply.

I.7.3.2.10 RBSP slice segment trailing bits syntax

The specifications in clause F.7.3.2.10 apply.

I.7.3.2.11 RBSP trailing bits syntax

The specifications in clause F.7.3.2.11 apply.

I.7.3.2.12 Byte alignment syntax

The specifications in clause F.7.3.2.12 apply.

I.7.3.3 Profile, tier and level syntax

The specifications in clause F.7.3.3 apply.

I.7.3.4 Scaling list data syntax

The specifications in clause F.7.3.4 apply.

I.7.3.5 Supplemental enhancement information message syntax

The specifications in clause F.7.3.5 apply.

I.7.3.6 Slice segment header syntax

I.7.3.6.1 General slice segment header syntax

slice_segment_header() {	Descriptor
first_slice_segment_in_pic_flag	u(1)
if(nal_unit_type >= BLA_W_LP && nal_unit_type <= RSV_IRAP_VCL23)	
no_output_of_prior_pics_flag	u(1)
slice_pic_parameter_set_id	ue(v)
if(!first_slice_segment_in_pic_flag) {	
if(dependent_slice_segments_enabled_flag)	
dependent_slice_segment_flag	u(1)
slice_segment_address	u(v)
}	
if(!dependent_slice_segment_flag) {	
i = 0	
if(num_extra_slice_header_bits > i) {	
i++	
discardable_flag	u(1)
}	
if(num_extra_slice_header_bits > i) {	
i++	
cross_layer_bla_flag	u(1)
}	
for(i < num_extra_slice_header_bits; i++)	
slice_reserved_flag[i]	u(1)
slice_type	ue(v)
if(output_flag_present_flag)	
pic_output_flag	u(1)
if(separate_colour_plane_flag == 1)	
colour_plane_id	u(2)
if((nuh_layer_id > 0 && !poc_lsb_not_present_flag[LayerIdxInVps[nuh_layer_id]]) (nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP))	
slice_pic_order_cnt_lsb	u(v)
if(nal_unit_type != IDR_W_RADL && nal_unit_type != IDR_N_LP) {	
short_term_ref_pic_set_sps_flag	u(1)
if(!short_term_ref_pic_set_sps_flag)	
st_ref_pic_set(num_short_term_ref_pic_sets)	
else if(num_short_term_ref_pic_sets > 1)	
short_term_ref_pic_set_idx	u(v)
if(long_term_ref_pics_present_flag) {	

if(num_long_term_ref_pics_sps > 0)	
num_long_term_sps	ue(v)
num_long_term_pics	ue(v)
for(i = 0; i < num_long_term_sps + num_long_term_pics; i++) {	
if(i < num_long_term_sps) {	
if(num_long_term_ref_pics_sps > 1)	
lt_idx_sps[i]	u(v)
} else {	
poc_lsb_lt[i]	u(v)
used_by_curr_pic_lt_flag[i]	u(1)
}	
delta_poc_msb_present_flag[i]	u(1)
if(delta_poc_msb_present_flag[i])	
delta_poc_msb_cycle_lt[i]	ue(v)
}	
}	
if(sps_temporal_mvp_enabled_flag)	
slice_temporal_mvp_enabled_flag	u(1)
}	
if(nuh_layer_id > 0 && !default_ref_layers_active_flag && NumRefListLayers[nuh_layer_id] > 0) {	
inter_layer_pred_enabled_flag	u(1)
if(inter_layer_pred_enabled_flag && NumRefListLayers[nuh_layer_id] > 1) {	
if(!max_one_active_ref_layer_flag)	
num_inter_layer_ref_pics_minus1	u(v)
if(NumActiveRefLayerPics != NumRefListLayers[nuh_layer_id])	
for(i = 0; i < NumActiveRefLayerPics; i++)	
inter_layer_pred_layer_idc[i]	u(v)
}	
}	
if(inCmpPredAvailFlag)	
in_comp_pred_flag	u(1)
if(sample_adaptive_offset_enabled_flag) {	
slice_sao_luma_flag	u(1)
if(ChromaArrayType != 0)	
slice_sao_chroma_flag	u(1)
}	
if(slice_type == P slice_type == B) {	
num_ref_idx_active_override_flag	u(1)
if(num_ref_idx_active_override_flag) {	
num_ref_idx_l0_active_minus1	ue(v)
if(slice_type == B)	
num_ref_idx_l1_active_minus1	ue(v)
}	
if(lists_modification_present_flag && NumPicTotalCurr > 1)	
ref_pic_lists_modification()	
if(slice_type == B)	
mvd_l1_zero_flag	u(1)
if(cabac_init_present_flag)	

cabac_init_flag	u(1)
if(slice_temporal_mvp_enabled_flag) {	
if(slice_type == B)	
collocated_from_l0_flag	u(1)
if((collocated_from_l0_flag && num_ref_idx_l0_active_minus1 > 0) (!collocated_from_l0_flag && num_ref_idx_l1_active_minus1 > 0))	
collocated_ref_idx	ue(v)
}	
if((weighted_pred_flag && slice_type == P) (weighted_bipred_flag && slice_type == B))	
pred_weight_table()	
else if(!DepthFlag && NumRefListLayers[nuh_layer_id] > 0) {	
slice_ic_enabled_flag	u(1)
if(slice_ic_enabled_flag)	
slice_ic_disabled_merge_zero_idx_flag	u(1)
}	
five_minus_max_num_merge_cand	ue(v)
}	
slice_qp_delta	se(v)
if(pps_slice_chroma_qp_offsets_present_flag) {	
slice_cb_qp_offset	se(v)
slice_cr_qp_offset	se(v)
}	
if(chroma_qp_offset_list_enabled_flag)	
cu_chroma_qp_offset_enabled_flag	u(1)
if(deblocking_filter_override_enabled_flag)	
deblocking_filter_override_flag	u(1)
if(deblocking_filter_override_flag) {	
slice_deblocking_filter_disabled_flag	u(1)
if(!slice_deblocking_filter_disabled_flag) {	
slice_beta_offset_div2	se(v)
slice_tc_offset_div2	se(v)
}	
}	
if(pps_loop_filter_across_slices_enabled_flag && (slice_sao_luma_flag slice_sao_chroma_flag !slice_deblocking_filter_disabled_flag))	
slice_loop_filter_across_slices_enabled_flag	u(1)
if(cp_in_slice_segment_header_flag[ViewIdx])	
for(m = 0; m < num_cp[ViewIdx]; m++) {	
j = cp_ref_voi[ViewIdx][m]	
cp_scale[j]	se(v)
cp_off[j]	se(v)
cp_inv_scale_plus_scale[j]	se(v)
cp_inv_off_plus_off[j]	se(v)
}	
}	
if(tiles_enabled_flag entropy_coding_sync_enabled_flag) {	
num_entry_point_offsets	ue(v)

if(num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset_minus1[i]	u(v)
}	
}	
if(slice_segment_header_extension_present_flag) {	
slice_segment_header_extension_length	ue(v)
if(poc_reset_info_present_flag)	
poc_reset_idc	u(2)
if(poc_reset_idc != 0)	
poc_reset_period_id	u(6)
if(poc_reset_idc == 3) {	
full_poc_reset_flag	u(1)
poc_lsb_val	u(v)
}	
if(!PocMsbValRequiredFlag && vps_poc_lsb_aligned_flag)	
poc_msb_cycle_val_present_flag	u(1)
if(poc_msb_cycle_val_present_flag)	
poc_msb_cycle_val	ue(v)
while(more_data_in_slice_segment_header_extension())	
slice_segment_header_extension_data_bit	u(1)
}	
byte_alignment()	
}	

I.7.3.6.2 Reference picture list modification syntax

The specifications in clause F.7.3.6.2 apply.

I.7.3.6.3 Weighted prediction parameters syntax

The specifications in clause F.7.3.6.3 apply.

I.7.3.7 Short-term reference picture set syntax

The specifications in clause F.7.3.7 apply.

I.7.3.8 Slice segment data syntax

I.7.3.8.1 General slice segment data syntax

The specifications in clause F.7.3.8.1 apply.

I.7.3.8.2 Coding tree unit syntax

The specifications in clause F.7.3.8.2 apply.

I.7.3.8.3 Sample adaptive offset syntax

The specifications in clause F.7.3.8.3 apply.

I.7.3.8.4 Coding quadtree syntax

	Descriptor
<code>coding_quadtree(x0, y0, log2CbSize, cqtDepth) {</code>	
<code> if(x0 + (1 << log2CbSize) <= pic_width_in_luma_samples && y0 + (1 << log2CbSize) <= pic_height_in_luma_samples && log2CbSize > MinCbLog2SizeY && !predSplitCuFlag)</code>	
<code> split_cu_flag[x0][y0]</code>	<code>ae(v)</code>
<code> if(cu_qp_delta_enabled_flag && log2CbSize >= Log2MinCuQpDeltaSize) {</code>	
<code> IsCuQpDeltaCoded = 0</code>	
<code> CuQpDeltaVal = 0</code>	
<code> }</code>	
<code> if(cu_chroma_qp_offset_enabled_flag && log2CbSize >= Log2MinCuChromaQpOffsetSize)</code>	
<code> IsCuChromaQpOffsetCoded = 0</code>	
<code> if(split_cu_flag[x0][y0]) {</code>	
<code> x1 = x0 + (1 << (log2CbSize - 1))</code>	
<code> y1 = y0 + (1 << (log2CbSize - 1))</code>	
<code> coding_quadtree(x0, y0, log2CbSize - 1, cqtDepth + 1)</code>	
<code> if(x1 < pic_width_in_luma_samples)</code>	
<code> coding_quadtree(x1, y0, log2CbSize - 1, cqtDepth + 1)</code>	
<code> if(y1 < pic_height_in_luma_samples)</code>	
<code> coding_quadtree(x0, y1, log2CbSize - 1, cqtDepth + 1)</code>	
<code> if(x1 < pic_width_in_luma_samples && y1 < pic_height_in_luma_samples)</code>	
<code> coding_quadtree(x1, y1, log2CbSize - 1, cqtDepth + 1)</code>	
<code> } else</code>	
<code> coding_unit(x0, y0, log2CbSize)</code>	
<code>}</code>	

I.7.3.8.5 Coding unit syntax

	Descriptor
<code>coding_unit(x0, y0, log2CbSize) {</code>	
<code> if(transquant_bypass_enabled_flag)</code>	
<code> cu_transquant_bypass_flag</code>	<code>ae(v)</code>
<code> if(slice_type != I)</code>	
<code> cu_skip_flag[x0][y0]</code>	<code>ae(v)</code>
<code> nCbS = (1 << log2CbSize)</code>	
<code> if(cu_skip_flag[x0][y0])</code>	
<code> prediction_unit(x0, y0, nCbS, nCbS)</code>	
<code> else if(SkipIntraEnabledFlag)</code>	
<code> skip_intra_flag[x0][y0]</code>	<code>ae(v)</code>
<code> if(!cu_skip_flag[x0][y0] && !skip_intra_flag[x0][y0]) {</code>	
<code> if(slice_type != I)</code>	
<code> pred_mode_flag</code>	<code>ae(v)</code>
<code> if((CuPredMode[x0][y0] != MODE_INTRA log2CbSize == MinCbLog2SizeY) && !predPartModeFlag)</code>	
<code> part_mode</code>	<code>ae(v)</code>
<code> if(CuPredMode[x0][y0] == MODE_INTRA) {</code>	

if(PartMode == PART_2Nx2N && pcm_enabled_flag && log2CbSize >= Log2MinIpcmCbSizeY && log2CbSize <= Log2MaxIpcmCbSizeY)	
pcm_flag [x0][y0]	ae(v)
if(pcm_flag[x0][y0]) {	
while(!byte_aligned())	
pcm_alignment_zero_bit	f(1)
pcm_sample(x0, y0, log2CbSize)	
} else {	
pbOffset = (PartMode == PART_NxN) ? (nCbS / 2) : nCbS	
log2PbSize = log2CbSize - ((PartMode == PART_NxN) ? 1 : 0)	
for(j = 0; j < nCbS; j = j + pbOffset)	
for(i = 0; i < nCbS; i = i + pbOffset) {	
if(IntraDcOnlyWedgeEnabledFlag IntraContourEnabledFlag)	
intra_mode_ext(x0 + i, y0 + j, log2PbSize)	
if(no_dim_flag[x0 + i][y0 + j])	
prev_intra_luma_pred_flag [x0 + i][y0 + j]	ae(v)
}	
for(j = 0; j < nCbS; j = j + pbOffset)	
for(i = 0; i < nCbS; i = i + pbOffset)	
if(no_dim_flag[x0 + i][y0 + j]) {	
if(prev_intra_luma_pred_flag[x0 + i][y0 + j])	
mpm_idx [x0 + i][y0 + j]	ae(v)
else	
rem_intra_luma_pred_mode [x0 + i][y0 + j]	ae(v)
}	
if(ChromaArrayType == 3)	
for(j = 0; j < nCbS; j = j + pbOffset)	
for(i = 0; i < nCbS; i = i + pbOffset)	
intra_chroma_pred_mode [x0 + 1][y0 + j]	ae(v)
else if(ChromaArrayType != 0)	
intra_chroma_pred_mode[x0][y0]	ae(v)
}	
} else {	
if(PartMode == PART_2Nx2N)	
prediction_unit(x0, y0, nCbS, nCbS)	
else if(PartMode == PART_2NxN) {	
prediction_unit(x0, y0, nCbS, nCbS / 2)	
prediction_unit(x0, y0 + (nCbS / 2), nCbS, nCbS / 2)	
} else if(PartMode == PART_Nx2N) {	
prediction_unit(x0, y0, nCbS / 2, nCbS)	
prediction_unit(x0 + (nCbS / 2), y0, nCbS / 2, nCbS)	
} else if(PartMode == PART_2NxN) {	
prediction_unit(x0, y0, nCbS, nCbS / 4)	
prediction_unit(x0, y0 + (nCbS / 4), nCbS, nCbS * 3 / 4)	
} else if(PartMode == PART_2NxND) {	
prediction_unit(x0, y0, nCbS, nCbS * 3 / 4)	
prediction_unit(x0, y0 + (nCbS * 3 / 4), nCbS, nCbS / 4)	
} else if(PartMode == PART_nLx2N) {	

prediction_unit(x0, y0, nCbS / 4, nCbS)	
prediction_unit(x0 + (nCbS / 4), y0, nCbS * 3 / 4, nCbS)	
} else if(PartMode == PART_nRx2N) {	
prediction_unit(x0, y0, nCbS * 3 / 4, nCbS)	
prediction_unit(x0 + (nCbS * 3 / 4), y0, nCbS / 4, nCbS)	
} else { /* PART_NxN */	
prediction_unit(x0, y0, nCbS / 2, nCbS / 2)	
prediction_unit(x0 + (nCbS / 2), y0, nCbS / 2, nCbS / 2)	
prediction_unit(x0, y0 + (nCbS / 2), nCbS / 2, nCbS / 2)	
prediction_unit(x0 + (nCbS / 2), y0 + (nCbS / 2), nCbS / 2, nCbS / 2)	
}	
}	
}	
cu_extension(x0, y0, log2CbSize)	
if(DcOnlyFlag[x0][y0] (!skip_intra_flag[x0][y0] && CuPredMode[x0][y0] == MODE_INTRA))	
depth_dcs(x0, y0, log2CbSize)	
if(!cu_skip_flag[x0][y0] && !skip_intra_flag[x0][y0] && !dc_only_flag[x0][y0] && !pcm_flag[x0][y0]) {	
if(CuPredMode[x0][y0] != MODE_INTRA && !(PartMode == PART_2Nx2N && merge_flag[x0][y0]))	
rqt_root_cbf	ae(v)
if(rqt_root_cbf) {	
MaxTrafoDepth = (CuPredMode[x0][y0] == MODE_INTRA ? (max_transform_hierarchy_depth_intra + IntraSplitFlag) : max_transform_hierarchy_depth_inter)	
transform_tree(x0, y0, x0, y0, log2CbSize, 0, 0)	
}	
}	
}	

I.7.3.8.5.1 Intra mode extension syntax

intra_mode_ext(x0, y0, log2PbSize) {	Descriptor
if(log2PbSize < 6)	
no_dim_flag [x0][y0]	ae(v)
if(!no_dim_flag[x0][y0] && IntraDcOnlyWedgeEnabledFlag && IntraContourEnabledFlag)	
depth_intra_mode_idx_flag [x0][y0]	ae(v)
if(!no_dim_flag[x0][y0] && !depth_intra_mode_idx_flag[x0][y0])	
wedge_full_tab_idx [x0][y0]	ae(v)
}	

I.7.3.8.5.2 Coding unit extension syntax

	Descriptor
cu_extension(x0, y0, log2CbSize) {	
if(skip_intra_flag[x0][y0])	
skip_intra_mode_idx [x0][y0]	ae(v)
else {	
if(!cu_skip_flag[x0][y0]) {	
if(DbbpEnabledFlag && DispAvailFlag && log2CbSize > 3 && (PartMode == PART_2NxN PartMode == PART_Nx2N))	
dbbp_flag [x0][y0]	ae(v)
if((CuPredMode[x0][y0] == MODE_INTRA ? IntraDcOnlyWedgeEnabledFlag : InterDcOnlyEnabledFlag) && PartMode == PART_2Nx2N)	
dc_only_flag [x0][y0]	ae(v)
}	
if(CuPredMode[x0][y0] != MODE_INTRA && PartMode == PART_2Nx2N) {	
if(IvResPredEnabledFlag && RpRefPicAvailFlag)	
iv_res_pred_weight_idx [x0][y0]	ae(v)
if(slice_ic_enabled_flag && icCuEnableFlag && iv_res_pred_weight_idx[x0][y0] == 0)	
illu_comp_flag [x0][y0]	ae(v)
}	
}	
}	

I.7.3.8.5.3 Depth DCs syntax

	Descriptor
depth_dcs(x0, y0, log2CbSize) {	
nCbs = (1 << log2CbSize)	
pbOffset = (PartMode == PART_NxN && CuPredMode[x0][y0] == MODE_INTRA) ? (nCbs / 2) : nCbs	
for(j = 0; j < nCbs; j = j + pbOffset)	
for(k = 0; k < nCbs; k = k + pbOffset)	
if(DimFlag[x0 + k][y0 + j] DcOnlyFlag[x0][y0]) {	
if(CuPredMode[x0][y0] == MODE_INTRA && DcOnlyFlag[x0][y0])	
depth_dc_present_flag [x0 + k][y0 + j]	ae(v)
dcNumSeg = DimFlag[x0 + k][y0 + j] ? 2 : 1	
if(depth_dc_present_flag[x0 + k][y0 + j])	
for(i = 0; i < dcNumSeg; i++) {	
depth_dc_abs [x0 + k][y0 + j][i]	ae(v)
if((depth_dc_abs[x0 + k][y0 + j][i] - dcNumSeg + 2) > 0)	
depth_dc_sign_flag [x0 + k][y0 + j][i]	ae(v)
}	
}	
}	
}	

I.7.3.8.6 Prediction unit syntax

The specifications in clause F.7.3.8.6 apply.

I.7.3.8.7 PCM sample syntax

The specifications in clause F.7.3.8.7 apply.

I.7.3.8.8 Transform tree syntax

The specifications in clause F.7.3.8.8 apply.

I.7.3.8.9 Motion vector difference coding syntax

The specifications in clause F.7.3.8.9 apply.

I.7.3.8.10 Transform unit syntax

The specifications in clause F.7.3.8.10 apply.

I.7.3.8.11 Residual coding syntax

The specifications in clause F.7.3.8.11 apply.

I.7.3.8.12 Cross-component prediction syntax

The specifications in clause F.7.3.8.12 apply.

I.7.3.8.13 Palette mode syntax

The specifications in clause F.7.3.8.13 apply.

I.7.3.8.14 Delta QP syntax

The specifications in clause F.7.3.8.14 apply.

I.7.3.8.15 Chroma QP offset syntax

The specifications in clause F.7.3.8.15 apply.

I.7.4 Semantics

I.7.4.1 General

I.7.4.2 NAL unit semantics

I.7.4.2.1 General NAL unit semantics

The specifications in clause F.7.4.2.1 apply.

I.7.4.2.2 NAL unit header semantics

The specifications in clause F.7.4.2.2 apply.

I.7.4.2.3 Encapsulation of an SODB within an RBSP (informative)

The specifications in clause F.7.4.2.3 apply.

I.7.4.2.4 Order of NAL units and association to coded pictures, access units, and coded video sequences

The specifications in clause F.7.4.2.4 and all its subclauses apply.

I.7.4.3 Raw byte sequence payloads, trailing bits, and byte alignment semantics

I.7.4.3.1 Video parameter set RBSP semantics

The specifications in clause F.7.4.3.1 apply with the following modifications and additions:

vps_extension2_flag equal to 0 specifies that no **vps_3d_extension()** syntax structure and no **vps_extension_data_flag** syntax elements are present in the VPS RBSP syntax structure. **vps_extension2_flag** equal to 1 specifies that the **vps_3d_extension()** syntax structure and **vps_extension_data_flag** syntax elements may be present in the VPS RBSP syntax structure. When **MaxLayersMinus1** is greater than 0, **vps_extension2_flag** shall be equal to 1.

vps_3d_extension_flag equal to 0 specifies that no **vps_3d_extension()** syntax structure is present in the VPS RBSP syntax structure. **vps_3d_extension_flag** equal to 1 specifies that the **vps_3d_extension()** syntax structure is present in the VPS RBSP syntax structure. When **MaxLayersMinus1** is greater than 0, **vps_3d_extension_flag** shall be equal to 1.

vps_3d_extension_alignment_bit_equal_to_one shall be equal to 1.

vps_extension3_flag equal to 0 specifies that no **vps_extension_data_flag** syntax elements are present in the VPS RBSP syntax structure. **vps_extension3_flag** shall be equal to 0 in bitstreams conforming to this version of this Specification. The value of 1 for **vps_extension3_flag** is reserved for future use by ITU-T | ISO/IEC. Decoders conforming to this version of this Specification shall ignore all data that follow the value 1 for **vps_extension3_flag** in a VPS RBSP.

vps_extension_data_flag may have any value. Its presence and value do not affect decoder conformance to profiles specified in Annexes A, G, H, or I. Decoders conforming to a profile specified in Annexes A, G, H, or I shall ignore all **vps_extension_data_flag** syntax elements.

1.7.4.3.1.1 Video parameter set extension semantics

The specifications in clause F.7.4.3.1.1 apply with the following additions and modifications:

direct_dependency_type[i][j] indicates the type of dependency between the layer **predLayer** with **nuh_layer_id** equal **layer_id_in_nuh**[i] and the layer **refLayer** with **nuh_layer_id** equal to **layer_id_in_nuh**[j]. ((**direct_dependency_type**[i][j] + 1) & 0x1) greater than 0 specifies that samples of **refLayer** may be used for inter-layer prediction of **predLayer**. ((**direct_dependency_type**[i][j] + 1) & 0x1) equal to 0 specifies that samples of **refLayer** are not used for inter-layer prediction of **predLayer**. ((**direct_dependency_type**[i][j] + 1) & 0x2) greater than 0 specifies that motion vectors of **refLayer** may be used for inter-layer prediction of **predLayer**. ((**direct_dependency_type**[i][j] + 1) & 0x2) equal to 0 specifies that motion vectors of **refLayer** are not used for inter-layer prediction of **predLayer**. ((**direct_dependency_type**[i][j] + 1) & 0x4) greater than 0 specifies that coding quadtree and coding unit partitioning information of **refLayer** may be used for inter-layer prediction of **predLayer**. ((**direct_dependency_type**[i][j] + 1) & 0x4) equal to 0 specifies that coding quadtree and coding unit partitioning information of **refLayer** are not used for inter-layer prediction of the **predLayer**.

The length of the **direct_dependency_type**[i][j] syntax element is **direct_dep_type_len_minus2** + 2 bits. Although the value of **direct_dependency_type**[i][j] shall be in the range of 0 to 2, inclusive, when **predLayer** conforms to a profile specified in Annexes A, G, or H, and in the range of 0 to 6, inclusive, when the **predLayer** conforms to a profile specified in Annex I, decoders shall allow values of **direct_dependency_type**[i][j] in the range of 0 to $2^{32} - 2$, inclusive, to appear in the syntax.

The list **ViewOIdxList**[idx] is derived as follows:

```

idx = 0
ViewOIdxList[ idx++ ] = 0
for( i = 1; i <= MaxLayersMinus1; i++ ) {
    newViewFlag = 1
    for( j = 0; j < i; j++ )
        if( ViewOrderIdx[ layer_id_in_nuh[ i ] ] == ViewOrderIdx[ layer_id_in_nuh[ j ] ] )
            newViewFlag = 0
    if( newViewFlag )
        ViewOIdxList[ idx++ ] = ViewOrderIdx[ i ]
}

```

(I-7)

The variables **NumRefListLayers**[iNuhLid] and **IdRefListLayer**[iNuhLid] are derived as follows:

```

for( i = 0; i <= MaxLayersMinus1; i++ ) {
    iNuhLid = layer_id_in_nuh[ i ]
    NumRefListLayers[ iNuhLid ] = 0
    for( j = 0; j < NumDirectRefLayers[ iNuhLid ]; j++ ) {
        jNuhLid = IdDirectRefLayer[ iNuhLid ][ j ]
        if( DepthLayerFlag[ iNuhLid ] == DepthLayerFlag[ jNuhLid ] )
            IdRefListLayer[ iNuhLid ][ NumRefListLayers[ iNuhLid ]++ ] = jNuhLid
    }
}

```

(I-8)

The variables **ViewCompLayerPresentFlag**[iViewOIdx][depFlag] and **ViewCompLayerId**[iViewOIdx][depFlag] are derived as follows:

```

for( depFlag = 0; depFlag <= 1; depFlag++ )
    for( i = 0; i < NumViews; i++ ) {
        iViewOIdx = ViewOIdxList[ i ]
        layerId = -1
        for( j = 0; j <= MaxLayersMinus1; j++ ) {
            jNuhLid = layer_id_in_nuh[ j ]
            if( DepthLayerFlag[ jNuhLid ] == depFlag && ViewOrderIdx[ jNuhLid ] == iViewOIdx
                && DependencyId[ jNuhLid ] == 0 && AuxId[ jNuhLid ] == 0 )
                layerId = jNuhLid
        }
    }

```

(I-9)


```

    }
    ViewCompLayerPresentFlag[ iViewOIdx ][ depFlag ] = ( layerId != -1 )
    ViewCompLayerId[ iViewOIdx ][ depFlag ] = layerId
}

```

The function ViewIdx(picX) is specified as follows:

$$\text{ViewIdx(picX)} = \text{ViewIdx of the picture picX} \quad (\text{I-10})$$

The function ViewIdVal(picX) is specified as follows:

$$\text{ViewIdVal(picX)} = \text{view_id_val[ViewIdx(picX)]} \quad (\text{I-11})$$

I.7.4.3.1.2 Representation format semantics

The specifications in clause F.7.4.3.1.2 apply.

I.7.4.3.1.3 DPB size semantics

The specifications in clause F.7.4.3.1.3 apply.

I.7.4.3.1.4 VPS VUI semantics

The specifications in clause F.7.4.3.1.4 apply.

I.7.4.3.1.5 Video signal info semantics

The specifications in clause F.7.4.3.1.5 apply.

I.7.4.3.1.6 VPS VUI bitstream partition HRD parameters semantics

The specifications in clause F.7.4.3.1.6 apply.

I.7.4.3.1.7 Video parameter set 3D extension semantics

cp_precision + BitDepth_V – 1 specifies the precision of the vps_cp_scale[i][j] and vps_cp_inv_scale_plus_scale[i][j] syntax elements present in the VPS and the cp_scale[j] and cp_inv_scale_plus_scale[j] syntax elements present in slice headers. The value of cp_precision shall be in the range of 0 to 5, inclusive.

num_cp[i], when cp_in_slice_segment_header_flag[i] is equal to 0, specifies the number of vps_cp_scale[i][j], vps_cp_off[i][j], vps_cp_inv_scale_plus_scale[i][j], and vps_cp_inv_off_plus_off[i][j] syntax elements present for the view with ViewIdx equal to i in the VPS. num_cp[i], when cp_in_slice_segment_header_flag[i] is equal to 1, specifies the number of cp_scale[j], cp_off[j], cp_inv_scale_plus_scale[j], and cp_inv_off_plus_off[j] syntax elements present in slice headers of layers with ViewIdx equal to i.

cp_in_slice_segment_header_flag[i] equal to 1 specifies that the syntax elements vps_cp_scale[i][j], vps_cp_off[i][j], vps_cp_inv_scale_plus_scale[i][j], and vps_cp_inv_off_plus_off[i][j] for the view with ViewIdx equal to i are not present in the VPS and that the syntax elements cp_scale[j], cp_off[j], cp_inv_scale_plus_scale[j], and cp_inv_off_plus_off[j] may be present in slice headers of layers with ViewIdx equal to i. cp_in_slice_segment_header_flag[i] equal to 0 specifies that the vps_cp_scale[i][j], vps_cp_off[i][j], vps_cp_inv_scale_plus_scale[i][j], and vps_cp_inv_off_plus_off[i][j] syntax elements for the view with ViewIdx equal to i are present in the VPS and that the syntax elements cp_scale[j], cp_off[j], cp_inv_scale_plus_scale[j], and cp_inv_off_plus_off[j] are not present in slice headers of layers with ViewIdx equal to i. When not present, the value of cp_in_slice_segment_header_flag[i] is inferred to be equal to 0.

cp_ref_voi[i][m], when cp_in_slice_segment_header_flag[i] is equal to 0, specifies the ViewIdx value j of the view to which the m-th vps_cp_scale[i][j], vps_cp_off[i][j], vps_cp_inv_scale_plus_scale[i][j], and vps_cp_inv_off_plus_off[i][j] syntax element present for the view with ViewIdx equal to i in the VPS is related to. cp_ref_voi[i][m], when cp_in_slice_segment_header_flag[i] is equal to 1, specifies the ViewIdx value j of the view to which the m-th cp_scale[j], cp_off[j], cp_inv_scale_plus_scale[j], and cp_inv_off_plus_off[j] syntax element present in the slice headers of layers with ViewIdx equal to i is related to. The value of cp_ref_voi[i][m] shall be in the range of 0 to 65 535, inclusive. It is a requirement of bitstream conformance that cp_ref_voi[i][x] is not equal to cp_ref_voi[i][y] for any values of x and y in the range of 0 to num_cp[i] – 1, inclusive, when x is not equal to y.

For n and m in the range of 0 to NumViews – 1, inclusive, the variable CpPresentFlag[ViewOIdxList[n]][ViewOIdxList[m]] is set equal to 0 and modified as follows:

```

for( n = 1; n < NumViews; n++ ) {
    i = ViewOIdxList[ n ]
    for( m = 0; m < num_cp[ i ]; m++ )
        CpPresentFlag[ i ][ cp_ref_voi[ i ][ m ] ] = 1
}

```

(I-12)

vps_cp_scale[i][j], **vps_cp_off**[i][j], **vps_cp_inv_scale_plus_scale**[i][j], and **vps_cp_inv_off_plus_off**[i][j] specify parameters for derivation of a horizontal component of a disparity vector from a depth value and may be used to infer the values of the **cp_scale**[j], **cp_off**[j], **cp_inv_scale_plus_scale**[j], and **cp_inv_off_plus_off**[j] syntax elements in slice headers of layers with ViewIdx equal to i. When both a texture layer and a depth layer with ViewIdx equal to i are present, the conversion parameters are associated with the texture layer with ViewIdx equal to i.

I.7.4.3.2 Sequence parameter set RBSP semantics

I.7.4.3.2.1 General sequence parameter set RBSP semantics

The specifications in clause F.7.4.3.2.1 apply.

I.7.4.3.2.2 Sequence parameter set range extension semantics

The specifications in clause F.7.4.3.2.2 apply.

I.7.4.3.2.1 Sequence parameter set screen content coding extension semantics

The specifications in clause F.7.4.3.2.3 apply.

I.7.4.3.2.2 Sequence parameter set multilayer extension semantic

The specifications in clause F.7.4.3.2.4 apply.

I.7.4.3.2.3 Sequence parameter set 3D extension semantics

iv_di_mc_enabled_flag[d] equal to 1 specifies that the derivation process for inter-view predicted merging candidates and the derivation process for disparity information merging candidates may be used in the decoding process of layers with DepthFlag equal to d. **iv_di_mc_enabled_flag**[d] equal to 0 specifies that derivation process for inter-view predicted merging candidates and the derivation process for disparity information merging candidates is not used in the decoding process of layers with DepthFlag equal to d. When not present, the value of **iv_di_mc_enabled_flag**[d] is inferred to be equal to 0.

iv_mv_scal_enabled_flag[d] equal to 1 specifies that motion vectors used for inter-view prediction may be scaled based on view_id_val values in the decoding process of layers with DepthFlag equal to d. **iv_mv_scal_enabled_flag**[d] equal to 0 specifies that motion vectors used for inter-view prediction are not scaled based on view_id_val values in the decoding process of layers with DepthFlag equal to d. When not present, the value of **iv_mv_scal_enabled_flag**[d] is inferred to be equal to 0.

log2_ivmc_sub_pb_size_minus3[d], when **iv_di_mc_enabled_flag**[d] is equal to 1 and d is equal to 0, is used to derive the minimum size of sub-block partitions used in the derivation process for sub-block partition motion vectors for an inter-layer predicted merging candidate in the decoding process of layers with DepthFlag equal to d. When not present, the value of **log2_ivmc_sub_pb_size_minus3**[d] is inferred to be equal to (CtbLog2SizeY – 3). The value of **log2_ivmc_sub_pb_size_minus3**[d] shall be in the range of (MinCbLog2SizeY – 3) to (CtbLog2SizeY – 3), inclusive.

iv_res_pred_enabled_flag[d] equal to 1 specifies that the **iv_res_pred_weight_idx** syntax element may be present in coding units of layers with DepthFlag equal to d. **iv_res_pred_enabled_flag**[d] equal to 0 specifies that the **iv_res_pred_weight_idx** syntax element is not present coding units of layers with DepthFlag equal to d. When not present, the value of **iv_res_pred_enabled_flag**[d] is inferred to be equal to 0.

vsp_mc_enabled_flag[d] equal to 1 specifies that the derivation process for a view synthesis prediction merging candidate may be used in the decoding process of layers with DepthFlag equal to d. **vsp_mc_enabled_flag**[d] equal to 0 specifies that the derivation process for a view synthesis prediction merging candidate is not used in the decoding process of layers with DepthFlag equal to d. When not present, the value of **vsp_mc_enabled_flag**[d] is inferred to be equal to 0.

dbbp_enabled_flag[d] equal to 1 specifies that the **dbbp_flag** syntax element may be present in coding units of layers with DepthFlag equal to d. **dbbp_enabled_flag**[d] equal to 0 specifies that the **dbbp_flag** syntax element is not present coding units of layers with DepthFlag equal to d. When not present, the value of **dbbp_enabled_flag**[d] is inferred to be equal to 0.

depth_ref_enabled_flag[d] equal to 1 specifies that the derivation process for a depth or disparity sample array from a depth picture may be used in the derivation process for a disparity vector for texture layers in the decoding process of layers with DepthFlag equal to d. **depth_ref_enabled_flag**[d] equal to 0 specifies that derivation process for a depth or disparity sample array from a depth picture is not used in the derivation process for a disparity vector for texture layers in the decoding process of layers with DepthFlag equal to d. When not present, the value of **depth_ref_enabled_flag**[d] is inferred to be equal to 0.

tex_mc_enabled_flag[d] equal to 1 specifies that the derivation process for motion vectors for the texture merge candidate may be used in the decoding process of layers with DepthFlag equal to d. **tex_mc_enabled_flag**[d] equal to 0

specifies that the derivation process for motion vectors for the texture merge candidate is not used in the decoding process of layers with DepthFlag equal to d. When not present, the value of `tex_mc_enabled_flag[d]` is inferred to be equal to 0.

`log2_texmc_sub_pb_size_minus3[d]`, when `tex_mc_enabled_flag[d]` is equal to 1, is used to derive the minimum size of sub-block partitions used in the derivation process for sub-block partition motion vectors for an inter-layer predicted merging candidate in the decoding process of layers with DepthFlag equal to d. The value of `log2_texmc_sub_pb_size_minus3[layerId]` shall be in the range of (`MinCbLog2SizeY - 3`) to (`CtbLog2SizeY - 3`), inclusive.

`intra_contour_enabled_flag[d]` equal to 1 specifies that the intra prediction mode INTRA_CONTOUR using depth intra contour prediction may be used in the decoding process of layers with DepthFlag equal to d. `intra_contour_enabled_flag[d]` equal to 0 specifies that the intra prediction mode INTRA_CONTOUR using depth intra contour prediction is not used in the decoding process of layers with DepthFlag equal to d. When not present, `intra_contour_enabled_flag[d]` is inferred to be equal to 0.

`intra_dc_only_wedge_enabled_flag[d]` equal to 1 specifies that the `dc_only_flag` syntax element may be present in coding units coded in an intra prediction mode of layers with DepthFlag equal to d, and that the intra prediction mode INTRA_WEDGE may be used in the decoding process of layers with DepthFlag equal to d. `intra_dc_only_wedge_enabled_flag[d]` equal to 0 specifies that the `dc_only_flag` syntax element is not present in coding units coded in an intra prediction mode of layers with DepthFlag equal to d and that the intra prediction mode INTRA_WEDGE is not used in the decoding process of layers with DepthFlag equal to d. When not present, the value of `intra_dc_only_wedge_enabled_flag[d]` is inferred to be equal to 0.

`cqt_cu_part_pred_enabled_flag[d]` equal to 1 specifies that coding quadtree and coding unit partitioning information may be inter-component predicted in the decoding process of layers with DepthFlag equal to d. `cqt_cu_part_pred_enabled_flag[d]` equal to 0 specifies that coding quadtree and coding unit partitioning information are not inter-component predicted in the decoding process of layers with DepthFlag equal to d. When not present, the value of `cqt_cu_part_pred_enabled_flag[d]` is inferred to be equal to 0.

`inter_dc_only_enabled_flag[d]` equal to 1 specifies that the `dc_only_flag` syntax element may be present in coding units coded in an inter prediction mode of layers with DepthFlag equal to d. `inter_dc_only_enabled_flag[d]` equal to 0 specifies that the `dc_only_flag` syntax element is not present in coding units coded in an inter prediction mode of layers with DepthFlag equal to d. When not present, the value of `inter_dc_only_enabled_flag[layerId]` is inferred to be equal to 0.

`skip_intra_enabled_flag[d]` equal to 1 specifies that the `skip_intra_flag` syntax element may be present in coding units of layers with DepthFlag equal to d. `skip_intra_enabled_flag[d]` equal to 0 specifies that the `skip_intra_flag` syntax element is not present in coding units of layers with DepthFlag equal to d. When not present, the value of `skip_intra_enabled_flag[layerId]` is inferred to be equal to 0.

I.7.4.3.3 Picture parameter set RBSP semantics

I.7.4.3.3.1 General picture parameter set RBSP semantics

The specifications in clause F.7.4.3.3.1 apply.

I.7.4.3.3.2 Picture parameter set range extension semantics

The specifications in clause F.7.4.3.3.2 apply.

I.7.4.3.3.3 Picture parameter set screen content coding extension semantics

The specifications in clause F.7.4.3.3.3 apply.

I.7.4.3.3.4 Picture parameter set multilayer extension semantics

The specifications in clause F.7.4.3.3.4 apply.

I.7.4.3.3.5 General colour mapping table semantics

The specifications in clause F.7.4.3.3.5 apply.

I.7.4.3.3.6 Colour mapping octants semantics

The specifications in clause F.7.4.3.3.6 apply.

I.7.4.3.3.7 Picture parameter set 3D extension semantics

`dlts_present_flag` equal to 1 specifies that syntax elements for the derivation of depth look-up tables are present in the PPS. `dlts_present_flag` equal to 0 specifies that syntax elements for the derivation of depth look-up tables are not present in the PPS.

The variables NumDepthLayers and DepIdxToLid[j] are derived as follows:

```

j = 0
for( i = 0; i <= MaxLayersMinus1; i++ ) {
    layerId = layer_id_in_nuh[ i ]
    if( DepthLayerFlag[ layerId ] )
        DepIdxToLid[ j++ ] = layerId
}
NumDepthLayers = j

```

(I-13)

pps_depth_layers_minus1 plus 1 specifies the number of depth layers. pps_depth_layers_minus1 shall be equal to NumDepthLayers – 1.

pps_bit_depth_for_depth_layers_minus8 plus 8 specifies the bit depth of the samples in depth layers. It is a requirement of bitstream conformance that pps_bit_depth_for_depth_layers_minus8 shall be equal to bit_depth_luma_minus8 of the SPS the current PPS refers to.

The variable depthMaxValue is set equal to $(1 \ll (\text{pps_bit_depth_for_depth_layers_minus8} + 8)) - 1$.

dlt_flag[i] equal to 1 specifies that a depth look-up table for the layer with nuh_layer_id equal to DepIdxToLid[i] is present in the PPS and used for the decoding of the layer with nuh_layer_id equal to DepIdxToLid[i]. dlt_flag[i] equal to 0 specifies that a depth look-up table is not present for the layer with nuh_layer_id equal to DepIdxToLid[i]. When not present, the value of dlt_flag[i] is inferred to be equal to 0.

For i in the range of 0 to NumDepthLayers – 1, inclusive, the variable DltFlag[DepIdxToLid[i]] is set equal to dlt_flag[i].

dlt_pred_flag[i] equal to 1 indicates that the depth look-up table of the layer with nuh_layer_id equal to DepIdxToLid[i] is predicted from the depth look-up table of the layer with nuh_layer_id equal to DepIdxToLid[0]. dlt_pred_flag[i] equal to 0 indicates that the depth look-up table of the layer with nuh_layer_id equal to DepIdxToLid[i] is not predicted from any other depth look-up table. The value of dlt_pred_flag[0] shall be equal to 0. It is a requirement of bitstream conformance that, when dlt_flag[0] is equal to 0, dlt_pred_flag[i] shall be equal to 0.

dlt_val_flags_present_flag[i] equal to 1 specifies the depth look-up table of the layer with nuh_layer_id equal to DepIdxToLid[i] is derived from dlt_value_flag[i][j] syntax elements. dlt_val_flags_present_flag[i] equal to 0 specifies the depth look-up table of the layer with nuh_layer_id equal to DepIdxToLid[i] is derived from the delta_dlt() syntax structure. When not present, the value of dlt_val_flags_present_flag[i] is inferred to be equal to 0.

dlt_value_flag[i][j] equal to 1 specifies that j is an entry in the depth look-up table of the layer with nuh_layer_id equal to DepIdxToLid[i]. dlt_value_flag[i][j] equal to 0 specifies that j is not an entry in the depth look-up table of the layer with nuh_layer_id equal to DepIdxToLid[i].

When dlt_val_flags_present_flag[i] is equal to 1, the following applies:

- The variable layerId is set equal to DepIdxToLid[i].
- The variables DltVal[layerId][n] and NumValDlt[layerId] of the depth look-up table of the layer with nuh_layer_id equal to layerId are derived as follows:

```

for( n = 0, j = 0; j <= depthMaxValue; j++ )
    if( dlt_value_flag[ i ][ j ] )
        DltVal[ layerId ][ n++ ] = j
NumValDlt[ layerId ] = n

```

(I-14)

1.7.4.3.3.8 Delta depth look-up table semantics

num_val_delta_dlt specifies the number of elements in the list deltaList. The length of num_val_delta_dlt syntax element is pps_bit_depth_for_depth_layers_minus8 + 8 bits.

max_diff specifies the maximum difference between two consecutive elements in the list deltaList. The length of max_diff syntax element is pps_bit_depth_for_depth_layers_minus8 + 8 bits. When not present, the value of max_diff is inferred to be equal to 0.

min_diff_minus1 specifies the minimum difference between two consecutive elements in the list deltaList. min_diff_minus1 shall be in the range of 0 to max_diff – 1, inclusive. The length of the min_diff_minus1 syntax element is $\text{Ceil}(\text{Log2}(\text{max_diff} + 1))$ bits. When not present, the value of min_diff_minus1 is inferred to be equal to (max_diff – 1).

The variable minDiff is set equal to (min_diff_minus1 + 1).

delta_dlt_val0 specifies the 0-th element in the list deltaList. The length of the delta_dlt_val0 syntax element is pps_bit_depth_for_depth_layers_minus8 + 8 bits.

delta_val_diff_minus_min[k] plus minDiff specifies the difference between the k-th element and the (k - 1)-th element in the list deltaList. The length of delta_val_diff_minus_min[k] syntax element is Ceil(Log2(max_diff - minDiff + 1)) bits. When not present, the value of delta_val_diff_minus_min[k] is inferred to be equal to 0.

The list deltaList is derived as follows:

```
deltaList[ 0 ] = delta_dlt_val0
for( k = 1; k < num_val_delta_dlt; k++ )
    deltaList[ k ] = deltaList[ k - 1 ] + delta_val_diff_minus_min[ k ] + minDiff
```

(I-15)

The variables layerId and refLayerId are set equal to DepIdxToLid[i] and DepIdxToLid[0], respectively.

The variables DltVal[layerId][n] and NumValDlt[layerId] of the depth look-up table of the layer with nuh_layer_id equal to layerId are derived as follows:

```
for( n = 0; j = 0; j <= depthMaxValue; j++ ) {
    inRefDltFlag = 0
    if( dlt_pred_flag[ i ] )
        for( k = 0; k < NumValDlt[ refLayerId ]; k++ )
            inRefDltFlag = inRefDltFlag || ( DltVal[ refLayerId ][ k ] == j )
    inUpdateDltFlag = 0
    for( k = 0; k < num_val_delta_dlt; k++ )
        inUpdateDltFlag = inUpdateDltFlag || ( deltaList[ k ] == j )
    if( inRefDltFlag != inUpdateDltFlag )
        DltVal[ layerId ][ n++ ] = j
}
NumValDlt[ layerId ] = n
```

(I-16)

I.7.4.3.4 Supplemental enhancement information RBSP semantics

The specifications in clause F.7.4.3.4 apply.

I.7.4.3.5 Access unit delimiter RBSP semantics

The specifications in clause F.7.4.3.5 apply.

I.7.4.3.6 End of sequence RBSP semantics

The specifications in clause F.7.4.3.6 apply.

I.7.4.3.7 End of bitstream RBSP semantics

The specifications in clause F.7.4.3.7 apply.

I.7.4.3.8 Filler data RBSP semantics

The specifications in clause F.7.4.3.8 apply.

I.7.4.3.9 Slice segment layer RBSP semantics

The specifications in clause F.7.4.3.9 apply.

I.7.4.3.10 RBSP slice segment trailing bits semantics

The specifications in clause F.7.4.3.10 apply.

I.7.4.3.11 RBSP trailing bits semantics

The specifications in clause F.7.4.3.11 apply.

I.7.4.3.12 Byte alignment semantics

The specifications in clause F.7.4.3.12 apply.

I.7.4.4 Profile, tier and level semantics

The specifications in clause F.7.4.4 apply.

I.7.4.5 Scaling list data semantics

The specifications in clause F.7.4.5 apply.

I.7.4.6 Supplemental enhancement information message semantics

The specifications in clause F.7.4.6 apply.

I.7.4.7 Slice segment header semantics

I.7.4.7.1 General slice segment header semantics

The specifications in clause F.7.4.7.1 apply with the following modifications and additions.

The variable `DepthFlag` is set equal to `DepthLayerFlag[nuh_layer_id]` and the variable `ViewIdx` is set equal to `ViewOrderIdx[nuh_layer_id]`.

The list `curCmpLIds` and the variable `numCurCmpLIds` are derived as follows:

`curCmpLIds = DepthFlag ? { nuh_layer_id } : RefPicLayerId`

`numCurCmpLIds = DepthFlag ? 1 : NumActiveRefLayerPics`

The list `inCmpRefViewIds[i]`, the variable `cpAvailableFlag`, and the variable `allRefCmpLayersAvailFlag` are derived as follows:

- The variables `cpAvailableFlag` and `allRefCmpLayersAvailFlag` are set equal to 1.
- For `i` in the range of 0 to `numCurCmpLIds – 1`, inclusive, the following applies:
 - The variable `inCmpRefViewIds[i]` is set equal to `ViewOrderIdx[curCmpLIds[i]]`.
 - When `CpPresentFlag[ViewIdx][inCmpRefViewIds[i]]` is equal to 0, `cpAvailableFlag` is set equal to 0.
 - The variable `refCmpCurLIdAvailFlag` is set equal to 0.
 - When `ViewCompLayerPresentFlag[inCmpRefViewIds[i]][!DepthFlag]` is equal to 1, the following applies:
 - The variable `j` is set equal to `LayerIdxInVps[ViewCompLayerId[inCmpRefViewIds[i]][!DepthFlag]]`.
 - When all of the following conditions are true, `refCmpCurLIdAvailFlag` is set equal to 1:
 - `direct_dependency_flag[LayerIdxInVps[nuh_layer_id]][j]` is equal to 1.
 - `sub_layers_vps_max_minus1[j]` is greater than or equal to `TemporalId`.
 - `TemporalId` is equal to 0 or `max_tid_il_ref_pics_plus1[j][LayerIdxInVps[nuh_layer_id]]` is greater than `TemporalId`.
 - When `refCmpCurLIdAvailFlag` is equal to 0, `allRefCmpLayersAvailFlag` is set equal to 0.

The variable `inCmpPredAvailFlag` is derived as follows:

- If `allRefCmpLayersAvailFlag` is equal to 0, `inCmpPredAvailFlag` is set equal to 0.
- Otherwise (`allRefCmpLayersAvailFlag` is equal to 1), the following applies:
 - If `DepthFlag` is equal to 0, the following applies:

$$\text{inCmpPredAvailFlag} = \text{vsp_mc_enabled_flag}[\text{DepthFlag}] \mid \mid \text{dbbp_enabled_flag}[\text{DepthFlag}] \mid \mid \text{depth_ref_enabled_flag}[\text{DepthFlag}] \quad (\text{I-17})$$

- Otherwise (`DepthFlag` is equal to 1), the following applies:

$$\text{inCmpPredAvailFlag} = \text{intra_contour_enabled_flag}[\text{DepthFlag}] \mid \mid \text{cqt_cu_part_pred_enabled_flag}[\text{DepthFlag}] \mid \mid \text{tex_mc_enabled_flag}[\text{DepthFlag}] \quad (\text{I-18})$$

in_comp_pred_flag equal to 0 specifies that reference pictures required for inter-component prediction of the current picture may not be present and that inter-component prediction of the current picture is disabled. **in_comp_pred_flag** equal to 1 specifies all reference pictures required for inter-component prediction of the current picture are present and that inter-component prediction of the current picture is enabled. When not present, the value of **in_comp_pred_flag** is inferred to be equal to 0.

When `in_comp_pred_flag` is equal to 1, the following applies for `i` in the range of 0 to `numCurCmpLIds - 1`, inclusive:

- It is a requirement of bitstream conformance that there is a picture in the DPB with `PicOrderCntVal` equal to the `PicOrderCntVal` of the current picture, and a `nuh_layer_id` value equal to `ViewCompLayerId[inCmpRefViewIds[i]][!DepthFlag]`.

The variables `IvDiMcEnabledFlag`, `IvMvScalEnabledFlag`, `IvResPredEnabledFlag`, `VspMcEnabledFlag`, `DbbpEnabledFlag`, `DepthRefEnabledFlag`, `TexMcEnabledFlag`, `IntraContourEnabledFlag`, `IntraDcOnlyWedgeEnabledFlag`, `CqtCuPartPredEnabledFlag`, `InterDcOnlyEnabledFlag`, `SkipIntraEnabledFlag` and `DisparityDerivationFlag` are derived as follows:

$$\text{IvDiMcEnabledFlag} = \text{NumRefListLayers}[\text{nuh_layer_id}] > 0 \ \&\& \ \text{iv_di_mc_enabled_flag}[\text{DepthFlag}] \quad (\text{I-19})$$

$$\text{IvMvScalEnabledFlag} = \text{iv_mv_scal_enabled_flag}[\text{DepthFlag}] \ \&\& \ \text{ViewIdx} \neq 0 \quad (\text{I-20})$$

$$\text{IvResPredEnabledFlag} = \text{NumRefListLayers}[\text{nuh_layer_id}] > 0 \ \&\& \ \text{iv_res_pred_enabled_flag}[\text{DepthFlag}] \quad (\text{I-21})$$

$$\text{VspMcEnabledFlag} = \text{NumRefListLayers}[\text{nuh_layer_id}] > 0 \ \&\& \ \text{vsp_mc_enabled_flag}[\text{DepthFlag}] \ \&\& \ \text{in_comp_pred_flag} \ \&\& \ \text{cpAvailableFlag} \quad (\text{I-22})$$

$$\text{DbbpEnabledFlag} = \text{dbbp_enabled_flag}[\text{DepthFlag}] \ \&\& \ \text{in_comp_pred_flag} \quad (\text{I-23})$$

$$\text{DepthRefEnabledFlag} = \text{depth_ref_enabled_flag}[\text{DepthFlag}] \ \&\& \ \text{in_comp_pred_flag} \ \&\& \ \text{cpAvailableFlag} \quad (\text{I-24})$$

$$\text{TexMcEnabledFlag} = \text{tex_mc_enabled_flag}[\text{DepthFlag}] \ \&\& \ \text{in_comp_pred_flag} \quad (\text{I-25})$$

$$\text{IntraContourEnabledFlag} = \text{intra_contour_enabled_flag}[\text{DepthFlag}] \ \&\& \ \text{in_comp_pred_flag} \quad (\text{I-26})$$

$$\text{IntraDcOnlyWedgeEnabledFlag} = \text{intra_dc_only_wedge_enabled_flag}[\text{DepthFlag}] \quad (\text{I-27})$$

$$\text{CqtCuPartPredEnabledFlag} = \text{cqt_cu_part_pred_enabled_flag}[\text{DepthFlag}] \ \&\& \ \text{in_comp_pred_flag} \ \&\& \ \text{slice_type} \neq \text{I} \ \&\& \ !(\text{nal_unit_type} \geq \text{BLA_W_LP} \ \&\& \ \text{nal_unit_type} \leq \text{RSV_IRAP_VCL23}) \quad (\text{I-28})$$

$$\text{InterDcOnlyEnabledFlag} = \text{inter_dc_only_enabled_flag}[\text{DepthFlag}] \quad (\text{I-29})$$

$$\text{SkipIntraEnabledFlag} = \text{skip_intra_enabled_flag}[\text{DepthFlag}] \quad (\text{I-30})$$

$$\text{DisparityDerivationFlag} = \text{IvDiMcEnabledFlag} \ || \ \text{IvResPredEnabledFlag} \ || \ \text{VspMcEnabledFlag} \ || \ \text{DbbpEnabledFlag} \quad (\text{I-31})$$

When `TexMcEnabledFlag` is equal to 1, or `CqtCuPartPredEnabledFlag` is equal to 1, or `IntraContourEnabledFlag` is equal to 1, let `TexturePic` be the picture in the current access unit with `nuh_layer_id` equal to `ViewCompLayerId[ViewIdx][0]`.

num_inter_layer_ref_pics_minus1 plus 1 specifies the number of pictures that may be used in decoding of the current picture for inter-layer prediction. The length of the `num_inter_layer_ref_pics_minus1` syntax element is $\text{Ceil}(\text{Log2}(\text{NumRefListLayers}[\text{nuh_layer_id}]))$ bits. The value of `num_inter_layer_ref_pics_minus1` shall be in the range of 0 to `NumRefListLayers[nuh_layer_id] - 1`, inclusive.

The variables `numRefLayerPics` and `refLayerPicIdx[j]` are derived as follows:

```
for( i = 0, j = 0; i < NumRefListLayers[ nuh_layer_id ]; i++ ) {
    refLayerIdx = LayerIdxInVps[ IdRefListLayer[ nuh_layer_id ][ i ] ]
    if( sub_layers_vps_max_minus1[ refLayerIdx ] >= TemporalId && ( TemporalId == 0 ||
        max_tid_il_ref_pics_plus1[ refLayerIdx ][ LayerIdxInVps[ nuh_layer_id ] ] > TemporalId ) )
        refLayerPicIdx[ j++ ] = i
    }
numRefLayerPics = j
```

The variable `NumActiveRefLayerPics` is derived as follows:

```
if( nuh_layer_id == 0 || numRefLayerPics == 0 )
    NumActiveRefLayerPics = 0
else if( default_ref_layers_active_flag )
    NumActiveRefLayerPics = numRefLayerPics
else if( !inter_layer_pred_enabled_flag )
    NumActiveRefLayerPics = 0
else if( max_one_active_ref_layer_flag || NumRefListLayers[ nuh_layer_id ] == 1 )
    NumActiveRefLayerPics = 1
else
    NumActiveRefLayerPics = num_inter_layer_ref_pics_minus1 + 1
```

All slices of a coded picture shall have the same value of `NumActiveRefLayerPics`.

inter_layer_pred_layer_idc[i] specifies the variable, **RefPicLayerId[i]**, representing the **nuh_layer_id** of the *i*-th picture that may be used by the current picture for inter-layer prediction. The length of the **inter_layer_pred_layer_idc[i]** syntax element is $\text{Ceil}(\text{Log2}(\text{NumRefListLayers}[\text{nuh_layer_id}]))$ bits. The value of **inter_layer_pred_layer_idc[i]** shall be in the range of 0 to $\text{NumRefListLayers}[\text{nuh_layer_id}] - 1$, inclusive. When *i* is greater than 0, **inter_layer_pred_layer_idc[i]** shall be greater than **inter_layer_pred_layer_idc[i - 1]**. When not present, the value of **inter_layer_pred_layer_idc[i]** is inferred to be equal to **refLayerPicIdc[i]**.

The variables **RefPicLayerId[i]** for all values of *i* in the range of 0 to $\text{NumActiveRefLayerPics} - 1$, inclusive, are derived as follows:

$$\begin{aligned} &\text{for}(i = 0, j = 0; i < \text{NumActiveRefLayerPics}; i++) \\ &\quad \text{RefPicLayerId}[i] = \text{IdRefListLayer}[\text{nuh_layer_id}][\text{inter_layer_pred_layer_idc}[i]] \end{aligned} \quad (\text{I-34})$$

The variable **NumExtraMergeCand** is derived as follows:

$$\text{NumExtraMergeCand} = \text{IvDiMcEnabledFlag} \mid \mid \text{TexMcEnabledFlag} \mid \mid \text{VspMcEnabledFlag} \quad (\text{I-35})$$

five_minus_max_num_merge_cand specifies the maximum number of merging motion vector prediction (MVP) candidates supported in the slice subtracted from $(5 + \text{NumExtraMergeCand})$.

The maximum number of merging MVP candidates, **MaxNumMergeCand** is derived as follows:

$$\text{MaxNumMergeCand} = 5 + \text{NumExtraMergeCand} - \text{five_minus_max_num_merge_cand} \quad (\text{I-36})$$

The value of **MaxNumMergeCand** shall be in the range of 1 to $(5 + \text{NumExtraMergeCand})$, inclusive.

slice_ic_enabled_flag equal to 1 specifies that the **illu_comp_flag[x0][y0]** syntax element may be present in coding units of the current slice. **slice_ic_enabled_flag** equal to 0 specifies that the **illu_comp_flag[x0][y0]** syntax element is not present in coding units of the current slice. When not present, the value of **slice_ic_enabled_flag** is inferred to be equal to 0.

slice_ic_disabled_merge_zero_idx_flag equal to 1 specifies that the **illu_comp_flag[x0][y0]** syntax element is not present in coding units of the current slice when **merge_flag[x0][y0]** is equal to 1 and **merge_idx[x0][y0]** is equal to 0. **slice_ic_disabled_merge_zero_idx_flag** equal to 0 specifies that **illu_comp_flag[x0][y0]** syntax element may be present in coding units of the current slice when **merge_flag[x0][y0]** is equal to 1 and **merge_idx[x0][y0]** is equal to 0. When not present, the value of **slice_ic_disabled_merge_zero_idx_flag** is inferred to be equal to 0.

cp_scale[j], **cp_off[j]**, **cp_inv_scale_plus_scale[j]**, and **cp_inv_off_plus_off[j]** specify parameters for the derivation of a horizontal component of a disparity vector from a depth value. When not present and **CpPresentFlag[ViewIdx][j]** is equal to 1, the values of **cp_scale[j]**, **cp_off[j]**, **cp_inv_scale_plus_scale[j]**, and **cp_inv_off_plus_off[j]** are inferred to be equal to **vps_cp_scale[ViewIdx][j]**, **vps_cp_off[ViewIdx][j]**, **vps_cp_inv_scale_plus_scale[ViewIdx][j]**, and **vps_cp_inv_off_plus_off[ViewIdx][j]**, respectively. It is a requirement of bitstream conformance, that the values of **cp_scale[j]**, **cp_off[j]**, **cp_inv_scale_plus_scale[j]**, and **cp_inv_off_plus_off[j]** in a slice header having a **ViewIdx** equal to **viewIdxA** and the values of **cp_scale[j]**, **cp_off[j]**, **cp_inv_scale_plus_scale[j]**, and **cp_inv_off_plus_off[j]** in a slice header having a **ViewIdx** equal to **viewIdxB** shall be the same, when **viewIdxA** is equal to **viewIdxB**.

The variable **DepthToDisparityB[j][d]** specifying the horizontal component of a disparity vector between the current view and the view with **ViewIdx** equal *j* corresponding to the depth value *d* in the view with **ViewIdx** equal to *j* and the variable **DepthToDisparityF[j][d]** specifying the horizontal component of a disparity vector between the view with **ViewIdx** equal *j* and the current view corresponding to the depth value *d* in the current view are derived as follows:

- The variable **log2Div** is set equal to $(\text{BitDepth}_Y - 1 + \text{cp_precision})$.
- For *d* in range of 0 to $((1 \ll \text{BitDepth}_Y) - 1)$, inclusive, the following applies:
 - For *m* in the range of 0 to $(\text{num_cp}[\text{ViewIdx}] - 1)$, inclusive, the following applies:

$$j = \text{cp_ref_voi}[\text{ViewIdx}][m] \quad (\text{I-37})$$

$$\text{offset} = (\text{cp_off}[j] \ll \text{BitDepth}_Y) + ((1 \ll \text{log2Div}) \gg 1) \quad (\text{I-38})$$

$$\text{scale} = \text{cp_scale}[j] \quad (\text{I-39})$$

$$\text{DepthToDisparityB}[j][d] = (\text{scale} * d + \text{offset}) \gg \text{log2Div} \quad (\text{I-40})$$

$$\text{invOffset} = ((\text{cp_inv_off_plus_off}[j] - \text{cp_off}[j]) \ll \text{BitDepth}_Y) + ((1 \ll \text{log2Div}) \gg 1) \quad (\text{I-41})$$

$$\text{invScale} = \text{cp_inv_scale_plus_scale}[j] - \text{cp_scale}[j] \quad (\text{I-42})$$

$$\text{DepthToDisparityF}[j][d] = (\text{invScale} * d + \text{invOffset}) \gg \text{log2Div} \quad (\text{I-43})$$

I.7.4.7.2 Reference picture list modification semantics

The specifications in clause F.7.4.7.2 apply.

I.7.4.7.3 Weighted prediction parameters semantics

The specifications in clause F.7.4.7.3 apply.

I.7.4.8 Short-term reference picture set semantics

The specifications in clause F.7.4.8 apply.

I.7.4.9 Slice segment data semantics

I.7.4.9.1 General slice segment data semantics

The specifications in clause F.7.4.9.1 apply.

I.7.4.9.2 Coding tree unit semantics

The specifications in clause F.7.4.9.2 apply.

I.7.4.9.3 Sample adaptive offset semantics

The specifications in clause F.7.4.9.3 apply.

I.7.4.9.4 Coding quadtree semantics

The specifications in clause F.7.4.9.4 apply with the following modifications:

The variable `predSplitCuFlag` specifying whether the `split_cu_flag[x0][y0]` syntax element is inter-component predicted is derived as follows:

- If `CqtCuPartPredEnabledFlag` is equal to 1, the following applies:
 - Let `colTextCu` be the coding unit containing the luma coding block covering the luma location (`x0`, `y0`) in the picture `TexturePic`.
 - The variable `log2TextCbSize` is set equal to `log2CbSize` of the coding unit `colTextCu`.
 - The variable `predSplitCuFlag` is set equal to (`log2CbSize <= log2TextCbSize`).
- Otherwise (`CqtCuPartPredEnabledFlag` is equal to 0), `predSplitCuFlag` is set equal to 0.

`split_cu_flag[x0][y0]` specifies whether a coding unit is split into coding units with half horizontal and vertical size. The array indices `x0`, `y0` specify the location (`x0`, `y0`) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

When `split_cu_flag[x0][y0]` is not present, the following applies:

- If `log2CbSize` is greater than `MinCbLog2SizeY` and `predSplitCuFlag` is equal to 0, the value of `split_cu_flag[x0][y0]` is inferred to be equal to 1.
- Otherwise (`log2CbSize` is equal to `MinCbLog2SizeY` or `predSplitCuFlag` is equal to 1), the value of `split_cu_flag[x0][y0]` is inferred to be equal to 0.

I.7.4.9.5 Coding unit semantics

The specifications in clause F.7.4.9.5 apply with the following modifications and additions:

`cu_skip_flag[x0][y0]` equal to 1 specifies that for the current coding unit, when decoding a P or B slice, no more syntax elements except the merging candidate index `merge_idx[x0][y0]`, the `iv_res_pred_weight_idx[x0][y0]` syntax element, and the `illu_comp_flag[x0][y0]` syntax element may be parsed after `cu_skip_flag[x0][y0]`. `cu_skip_flag[x0][y0]` equal to 0 specifies that the coding unit is not skipped. The array indices `x0`, `y0` specify the location (`x0`, `y0`) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

When `cu_skip_flag[x0][y0]` is not present, it is inferred to be equal to 0.

`skip_intra_flag[x0][y0]` equal to 1 specifies that for the current coding unit no more syntax elements except `skip_intra_mode_idx[x0][y0]` are parsed after `skip_intra_flag[x0][y0]`. `skip_intra_flag[x0][y0]` equal to 0 specifies that more syntax elements may be parsed after `skip_intra_flag[x0][y0]`. When not present, the value of `skip_intra_flag[x0][y0]` is inferred to be equal to 0.

pred_mode_flag equal to 0 specifies that the current coding unit is coded in inter prediction mode. **pred_mode_flag** equal to 1 specifies that the current coding unit is coded in intra prediction mode. The variable $\text{CuPredMode}[x][y]$ is derived as follows for $x = x0..x0 + nCbS - 1$ and $y = y0..y0 + nCbS - 1$:

- If **pred_mode_flag** is equal to 0, $\text{CuPredMode}[x][y]$ is set equal to **MODE_INTER**.
- Otherwise (**pred_mode_flag** is equal to 1), $\text{CuPredMode}[x][y]$ is set equal to **MODE_INTRA**.

When **pred_mode_flag** is not present, the variable $\text{CuPredMode}[x][y]$ is derived as follows for $x = x0..x0 + nCbS - 1$ and $y = y0..y0 + nCbS - 1$:

- If **slice_type** is equal to I or **skip_intra_flag**[$x0][y0]$ is equal to 1, $\text{CuPredMode}[x][y]$ is inferred to be equal to **MODE_INTRA**.
- Otherwise (**slice_type** is equal to P or B and **skip_intra_flag**[$x0][y0]$ is equal to 0), when **cu_skip_flag**[$x0][y0]$ is equal to 1, $\text{CuPredMode}[x][y]$ is inferred to be equal to **MODE_SKIP**.

The variables **predPartModeFlag** and **partPredIdc** are derived as follows:

- If **CqtCuPartPredEnabledFlag** is equal to 1, the following applies:
 - Let **colTextCu** be the coding unit containing the luma coding block covering the luma location ($x0, y0$) in the picture **TexturePic**.
 - The variables **log2TextCbSize** and **partTextMode** are set equal to **log2CbSize** and **PartMode**, respectively, of the coding unit **colTextCu**.
 - The variable **predPartModeFlag** is derived as follows:

$$\text{predPartModeFlag} = \text{log2TextCbSize} == \text{log2CbSize} \ \&\& \ \text{partTextMode} == \text{PART_2Nx2N} \quad (\text{I-44})$$
 - The variable **partPredIdc** is derived as follows:
 - If one or more of the following conditions are true, **partPredIdc** is set equal to 0:
 - **log2TextCbSize** is not equal to **log2CbSize**.
 - **partTextMode** is equal to **PART_2Nx2N** or **PART_NxN**.
 - Otherwise, if **partTextMode** is equal to **PART_2NxN**, **PART_2NxN_U**, or **PART_2NxN_L**, **partPredIdc** is set equal to 1.
 - Otherwise, **partPredIdc** is set equal to 2.
- Otherwise (**CqtCuPartPredEnabledFlag** is equal to 0), **predPartModeFlag** and **partPredIdc** are set equal to 0.

part_mode specifies partitioning mode of the current coding unit. The semantics of **part_mode** depend on $\text{CuPredMode}[x0][y0]$. The variables **PartMode** and **IntraSplitFlag** are derived from the value of **part_mode** and **partPredIdc** as defined in Table I.1

Table I.1 – Name association to prediction mode and partitioning type

CuPredMode [$x0$][$y0$]	part_mode	partPredIdc	IntraSplitFlag	PartMode
------------------------------------	------------------	--------------------	-----------------------	-----------------

MODE_INTRA	0	0	0	PART_2Nx2N
	1	0	1	PART_NxN
MODE_INTER	0	0	0	PART_2Nx2N
	1	0	0	PART_2NxN
	2	0	0	PART_Nx2N
	3	0	0	PART_NxN
	4	0	0	PART_2NxN
	5	0	0	PART_2NxN
	6	0	0	PART_nLx2N
	7	0	0	PART_nRx2N
	0	1	0	PART_2Nx2N
	1	1	0	PART_2NxN
	2	1	0	PART_2NxN
	3	1	0	PART_2NxN
	0	2	0	PART_2Nx2N
	1	2	0	PART_Nx2N
	2	2	0	PART_nLx2N
	3	2	0	PART_nRx2N

The value of part_mode is restricted as follows:

- If CuPredMode[x0][y0] is equal to MODE_INTRA, part_mode shall be equal to 0 or 1.
- Otherwise (CuPredMode[x0][y0] is equal to MODE_INTER), the following applies:
 - If partPredIdc is equal to 0, the following applies:
 - If log2CbSize is greater than MinCbLog2SizeY and amp_enabled_flag is equal to 1, part_mode shall be in the range of 0 to 2, inclusive, or in the range of 4 to 7, inclusive.
 - Otherwise, if log2CbSize is greater than MinCbLog2SizeY and amp_enabled_flag is equal to 0, or log2CbSize is equal to 3, part_mode shall be in the range of 0 to 2, inclusive.
 - Otherwise (log2CbSize is greater than 3 and less than or equal to MinCbLog2SizeY), the value of part_mode shall be in the range of 0 to 3, inclusive.
 - Otherwise (partPredIdc is not equal to 0), the following applies:
 - If log2CbSize is greater than MinCbLog2SizeY and amp_enabled_flag is equal to 1, part_mode shall be in the range of 0 to 3, inclusive.
 - Otherwise (log2CbSize is equal to MinCbLog2SizeY or amp_enabled_flag is equal to 0), part_mode shall be in the range of 0 to 1, inclusive.

When part_mode is not present, the variables PartMode and IntraSplitFlag are derived as follows:

- PartMode is set equal to PART_2Nx2N.
- IntraSplitFlag is set equal to 0.

rqt_root_cbf equal to 1 specifies that the transform_tree() syntax structure is present for the current coding unit. **rqt_root_cbf** equal to 0 specifies that the transform_tree() syntax structure is not present for the current coding unit. When not present, the value of rqt_root_cbf is inferred to be equal to !DcOnlyFlag[x0][y0].

I.7.4.9.5.1 Intra mode extension semantics

no_dim_flag[x0][y0] equal to 1 specifies that the intra modes INTRA_WEDGE or INTRA_CONTOUR are not used for the current prediction unit. **no_dim_flag**[x0][y0] equal to 0 specifies that the intra mode INTRA_WEDGE or INTRA_CONTOUR is used for the current prediction unit. When not present, the value of no_dim_flag[x0][y0] is inferred to be equal to 1. When log2CbSize is greater than MaxTbLog2SizeY and DcOnlyFlag[x0][y0] is equal to 0, the value of no_dim_flag[x0][y0] shall be equal to 1.

For $x = x0..x0 + (1 \ll \log2PbSize) - 1$, $y = y0..y0 + (1 \ll \log2PbSize) - 1$, the variable DimFlag[x][y] is derived as follows:

$$\text{DimFlag}[x][y] = \text{!no_dim_flag}[x0][y0] \quad (\text{I-45})$$

depth_intra_mode_idx_flag[x0][y0] equal to 0, when **DimFlag**[x0][y0] is equal to 1, specifies that the intra mode INTRA_WEDGE is used for the current prediction unit. **depth_intra_mode_idx_flag**[x0][y0] equal to 1, when **DimFlag**[x0][y0] is equal to 1, specifies that the intra mode INTRA_CONTOUR is used for the current prediction unit. When not present, the value of **depth_intra_mode_idx_flag**[x0][y0] is inferred to be equal to (**!IntraDcOnlyWedgeEnabledFlag** || **IntraContourEnabledFlag**). When **DimFlag**[x0][y0] is equal to 1 and **nal_unit_type** is equal to BLA_W_LP, BLA_W_RADL, BLA_N_LP, IDR_W_RADL or IDR_N_LP, it is a requirement of bitstream conformance that **depth_intra_mode_idx_flag**[x0][y0] is equal to 0.

wedge_full_tab_idx[x0][y0] specifies the index of the partition pattern in the list **WedgePatternTable**[log2PbSize] when the intra mode INTRA_WEDGE is used for the current prediction unit.

1.7.4.9.5.2 Coding unit extension semantics

skip_intra_mode_idx[x0][y0] equal to 0, when **skip_intra_flag**[x0][y0] is equal to 1, specifies that the intra prediction mode INTRA_ANGULAR26 is used for the current prediction unit. **skip_intra_mode_idx**[x0][y0] equal to 1, when **skip_intra_flag**[x0][y0] is equal to 1, specifies that the intra prediction mode INTRA_ANGULAR10 is used for the current prediction unit. **skip_intra_mode_idx**[x0][y0] equal to 2 or 3, when **skip_intra_flag**[x0][y0] is equal to 1, specifies that the intra prediction mode INTRA_SINGLE is used for the current prediction unit.

dbbp_flag[x0][y0] equal to 1 specifies that the decoding process for inter sample prediction for depth predicted sub-block partitions is used for the current coding unit. **dbbp_flag**[x0][y0] equal to 0 specifies that the decoding process for inter sample prediction for depth predicted sub-block partitions is not used for the current coding unit. When not present, the value of **dbbp_flag**[x0][y0] is inferred to be equal to 0.

For $x = x0..x0 + (1 \ll \log2CbSize) - 1$, $y = y0..y0 + (1 \ll \log2CbSize) - 1$, the variable **DbbpFlag**[x][y] is derived as follows:

$$\text{DbbpFlag}[x][y] = \text{dbbp_flag}[x0][y0] \quad (\text{I-46})$$

dc_only_flag[x0][y0] equal to 1 specifies that the **transform_tree()** syntax structure is not present for the current coding unit and that the **depth_dcs()** syntax structure is present for the current coding unit. **dc_only_flag**[x0][y0] equal to 0 specifies the **transform_tree()** syntax structure may be present for the current coding unit and that the **depth_dcs()** syntax structure may be present for the current coding unit. When not present, the value of **dc_only_flag**[x0][y0] is inferred to be equal to 0. It is a requirement of bitstream conformance, that when **pcm_flag**[x0][y0] is equal to 1, the value of **dc_only_flag**[x0][y0] shall be equal to 0.

For $x = x0..x0 + (1 \ll \log2CbSize) - 1$, $y = y0..y0 + (1 \ll \log2CbSize) - 1$, the variable **DcOnlyFlag**[x][y] is derived as follows:

$$\text{DcOnlyFlag}[x][y] = \text{dc_only_flag}[x0][y0] \quad (\text{I-47})$$

iv_res_pred_weight_idx[x0][y0] not equal to 0 specifies that the bilinear sample interpolation and residual prediction process is used for the current coding unit and the index of the weighting factor used in the bilinear sample interpolation and residual prediction process. **iv_res_pred_weight_idx**[x0][y0] equal to 0 specifies that the bilinear sample interpolation and residual prediction process is not used for the current coding unit. When not present, the value of **iv_res_pred_weight_idx**[x0][y0] is inferred to be equal to 0.

When **CuPredMode**[x0][y0] is not equal to **MODE_INTRA**, the variable **icCuEnableFlag** is derived as follows:

- If **merge_flag**[x0][y0] is equal to 1, the following applies:

$$\text{icCuEnableFlag} = (\text{merge_idx}[x0][y0] \neq 0) \mid \mid \text{!slice_ic_disabled_merge_zero_idx_flag} \quad (\text{I-48})$$

- Otherwise (**merge_flag**[x0][y0] is equal to 0), the following applies:

- For X in the range of 0 to 1, inclusive, the variable **refViewIdxLX** is set equal to **ViewIdx(RefPicListX[ref_idx_IX[x0][y0]])**.
- The flag **icCuEnableFlag** is derived as follows:

$$\text{icCuEnableFlag} = (\text{inter_pred_idx}[x0][y0] \neq \text{Pred_L0} \ \&\& \ \text{refViewIdxL1} \neq \text{ViewIdx}) \mid \mid (\text{inter_pred_idx}[x0][y0] \neq \text{Pred_L1} \ \&\& \ \text{refViewIdxL0} \neq \text{ViewIdx}) \quad (\text{I-49})$$

illu_comp_flag[x0][y0] equal to 1 specifies that the illumination compensated sample prediction process is used for the current coding unit. **illu_comp_flag**[x0][y0] equal to 0 specifies that the illumination compensated sample prediction process is not used for the current coding unit. When not present, the value of **illu_comp_flag**[x0][y0] is inferred to be equal to 0.

For $x = x0..x0 + (1 \ll \log2CbSize) - 1$, $y = y0..y0 + (1 \ll \log2CbSize) - 1$, the variable **IlluCompFlag**[x][y] is

derived as follows:

$$\text{IlluCompFlag}[x][y] = \text{illu_comp_flag}[x0][y0] \quad (\text{I-50})$$

I.7.4.9.5.3 Depth DCs semantics

depth_dc_present_flag[$x0+k$][$y0+j$] equal to 1 specifies that the **depth_dc_abs**[$x0+k$][$y0+j$][i] syntax element is present and that the **depth_dc_sign_flag**[$x0+k$][$y0+j$][i] syntax element may be present. **depth_dc_present_flag**[$x0+k$][$y0+j$] equal to 0 specifies that the **depth_dc_abs**[$x0+k$][$y0+j$][i] and **depth_dc_sign_flag**[$x0+k$][$y0+j$][i] syntax elements are not present. When not present, the value of **depth_dc_present_flag**[$x0+k$][$y0+j$] is inferred to be equal to 1.

depth_dc_abs[$x0+k$][$y0+j$][i] and **depth_dc_sign_flag**[$x0+k$][$y0+j$][i] are used to derive **DcOffset**[$x0+k$][$y0+j$][i]. When not present, the values of **depth_dc_abs**[$x0+k$][$y0+j$][i] and **depth_dc_sign_flag**[$x0+k$][$y0+j$][i] are inferred to be equal to 0. The variable **DcOffset**[$x0+k$][$y0+j$][i] is derived as follows:

$$\text{DcOffset}[x0+k][y0+j][i] = (1 - 2 * \text{depth_dc_sign_flag}[x0+k][y0+j][i]) * (\text{depth_dc_abs}[x0+k][y0+j][i] - \text{dcNumSeg} + 2) \quad (\text{I-51})$$

I.7.4.9.6 Prediction unit semantics

The specifications in clause F.7.4.9.6 apply with the following addition at the end of the specification of semantics of **inter_pred_idc**[$x0$][$y0$]:

It is a requirement of bitstream conformance that, when **DbbpFlag**[$x0$][$y0$] is equal to 1, **inter_pred_idc**[$x0$][$y0$] shall not be equal to **PRED_BI**.

I.7.4.9.7 PCM sample semantics

The specifications in clause F.7.4.9.7 apply.

I.7.4.9.8 Transform tree semantics

The specifications in clause F.7.4.9.8 apply with the following additions at the end of the specification of **split_transform_flag**[$x0$][$y0$][**trafoDepth**]:

When **DimFlag**[$x0$][$y0$] is equal to 1 and **PartMode** is equal to **PART_2Nx2N**, the value of **split_transform_flag**[$x0$][$y0$][0] shall be equal to 0.

When **DimFlag**[$x0$][$y0$] is equal to 1 and **PartMode** is equal to **PART_NxN**, the value of **split_transform_flag**[$x0$][$y0$][1] shall be equal to 0.

I.7.4.9.9 Motion vector difference coding semantics

The specifications in clause F.7.4.9.9 apply.

I.7.4.9.10 Transform unit semantics

The specifications in clause F.7.4.9.10 apply.

I.7.4.9.11 Residual coding semantics

The specifications in clause F.7.4.9.11 apply.

I.7.4.9.12 Cross-component prediction semantics

The specifications in clause F.7.4.9.12 apply.

I.7.4.9.13 Palette mode semantics

The specifications in clause F.7.4.9.13 apply.

I.7.4.9.14 Delta QP semantics

The specifications in clause F.7.4.9.14 apply.

I.7.4.9.15 Chroma QP offset semantics

The specifications in clause F.7.4.9.15 apply.

I.8 Decoding process

I.8.1 General decoding process

I.8.1.1 General

The specifications in clause F.8.1.1 apply.

I.8.1.2 Decoding process for a coded picture with nuh_layer_id greater than 0

The decoding process for the current picture CurrPic is as follows:

1. The decoding of NAL units is specified in clause I.8.2.
2. The processes in clauses G.8.1.3, F.8.3.4 and I.8.3.1 to I.8.3.5 specify the following decoding processes using syntax elements in the slice segment layer and above:
 - Prior to decoding the first slice of the current picture, clause G.8.1.3 is invoked.
 - At the beginning of the decoding process for each P or B slice, the decoding process for reference picture lists construction specified in clause F.8.3.4 is invoked for derivation of reference picture list 0 (RefPicList0), and when decoding a B slice, reference picture list 1 (RefPicList1).
 - When DisparityDerivationFlag is equal to 1 and DepthFlag is equal to 0, the derivation process for the candidate picture list for disparity vector derivation in clause I.8.3.1 is invoked at the beginning of the decoding process for each P or B slice.
 - The variable DefaultRefViewIdx is set equal to –1 and the variable DispAvailFlag is set equal to 0.
 - When DisparityDerivationFlag is equal to 1, the derivation process for the default reference view order index for disparity derivation as specified in clause I.8.3.2 is invoked at the beginning of the decoding process for each P or B slice.
 - When DltFlag[nuh_layer_id] is equal to 1, the derivation process for a depth look-up table in clause I.8.3.3 is invoked at the beginning of the decoding process of the first slice segment of a coded picture.
 - At the beginning of the decoding process for each P or B slice, the derivation process for the alternative target reference index for temporal motion vector prediction in merge mode as specified in clause I.8.3.4 is invoked.
 - When IvResPredEnabledFlag is equal to 1, the derivation process for the target reference index for residual prediction as specified in clause I.8.3.5 is invoked at the beginning of the decoding process for each P or B slice.
3. The processes in clauses I.8.4, I.8.5, I.8.6, and I.8.7, specify decoding processes using syntax elements in all syntax structure layers. It is a requirement of bitstream conformance that the coded slices of the picture shall contain slice segment data for every CTU of the picture, such that the division of the picture into slices, the division of the slices into slice segments, and the division of the slice segments into CTUs each form a partitioning of the picture.

I.8.2 NAL unit decoding process

The specifications in clause G.8.2 apply.

I.8.3 Slice decoding process

I.8.3.1 Derivation process for the candidate picture list for disparity vector derivation

The function tempId(picA) is specified as follows:

$$\text{tempId}(\text{picA}) = \text{TemporalId of picA} \quad (\text{I-52})$$

The variable NumDdvCandPics is set equal to 0 and when slice_temporal_mvp_enabled_flag is equal to 1, the list DdvCandPicList is constructed as follows:

1. The variable X is set equal to (1 – collocated_from_l0_flag), the variable DdvCandPicList[0] is set equal to RefPicListX[collocated_ref_idx], and NumDdvCandPics is set equal to 1.
2. The variables NumDdvCandPics, DdvCandPicList[1], and minTempIdRefs are derived as follows:

```
minTempIdRefs = 7
for( k = 0; k <= ( slice_type == B ? 1 : 0 ); k++ ) {
    X = k ? collocated_from_l0_flag : ( 1 – collocated_from_l0_flag )
    for( i = 0; i <= num_ref_idx_lX_active_minus1; i++ )
```

```

    if( ViewIdx == ViewIdx( RefPicListX[ i ] )
        && NumDdvCandPics != 2
        && ( X == collocated_from_l0_flag || i != collocated_ref_idx ) )
    if( RefPicListX[ i ] is an IRAP picture )
        DdvCandPicList[ NumDdvCandPics++ ] = RefPicListX[ i ]
    else
        minTempIdRefs = Min( minTempIdRefs, tempId( RefPicListX[ i ] ) )
}

```

(I-53)

3. When NumDdvCandPics is equal to 1, the following applies:

```

minPocDiff = 255
for( k = 0; k <= ( slice_type == B ? 1 : 0 ); k++ ) {
    X = k ? collocated_from_l0_flag : ( 1 - collocated_from_l0_flag )
    for( i = 0; i <= num_ref_idx_lX_active_minus1; i++ )
        if( ViewIdx == ViewIdx( RefPicListX[ i ] )
            && ( X == collocated_from_l0_flag || i != collocated_ref_idx )
            && tempId( RefPicListX[ i ] ) == minTempIdRefs
            && Abs( DiffPicOrderCnt( CurrPic, RefPicListX[ i ] ) ) < minPocDiff ) {
                minPocDiff = Abs( DiffPicOrderCnt( CurrPic, RefPicListX[ i ] ) )
                Z = X
                candIdx = i
        }
    }
if( minPocDiff < 255 )
    DdvCandPicList[ NumDdvCandPics++ ] = RefPicListZ[ candIdx ]

```

(I-54)

I.8.3.2 Derivation process for the default reference view order index for disparity derivation

This process is invoked when the current slice is a P or B slice.

The following applies for candViewIdx in the range of 0 to (ViewIdx - 1), inclusive:

- The following applies for X in the range of 0 to (slice_type == B) ? 1 : 0, inclusive:
 - The following applies for i in the range of 0 to num_ref_idx_lX_active_minus1, inclusive:
 - When all of the following conditions are true, DefaultRefViewIdx is set equal to candViewIdx and DispAvailFlag is set equal to 1.
 - DispAvailFlag is equal to 0.
 - ViewIdx(RefPicListX[i]) is equal to candViewIdx.
 - DiffPicOrderCnt(CurrPic, RefPicListX[i]) is equal to 0.

When DepthFlag is equal to 1, DisparityDerivationFlag is equal to 1, and DispAvailFlag is equal to 1, it is a requirement of bitstream conformance that CpPresentFlag[ViewIdx][DefaultRefViewIdx] shall be equal to 1.

I.8.3.3 Derivation process for a depth look-up table

For i in the range of 0 to (1 << BitDepth_Y) - 1, inclusive, the list DltIdxToVal[i] specifying the depth value corresponding to the i-th index in the depth look-up table is derived as follows:

$$\text{DltIdxToVal}[i] = (i < \text{NumValDlt}[\text{nuh_layer_id}]) ? \text{DltVal}[\text{nuh_layer_id}][i] : 0 \quad (\text{I-55})$$

The list DltValToIdx[d] specifying the index in the depth look-up table corresponding to the depth value d is derived as follows:

```

for( d = 0; d < ( 1 << BitDepthY ); d++ ) {
    idxLower = 0
    foundFlag = 0
    for( iL = 1; iL < NumValDlt[ nuh_layer_id ]; iL++ )
        if( !foundFlag && DltIdxToVal[ iL ] > d ) {
            idxLower = iL - 1
            foundFlag = 1
        }
    idxUpper = foundFlag ? ( idxLower + 1 ) : ( NumValDlt[ nuh_layer_id ] - 1 )
    if( Abs( d - DltIdxToVal[ idxLower ] ) < Abs( d - DltIdxToVal[ idxUpper ] ) )
        DltValToIdx[ d ] = idxLower
}

```

(I-56)

```

else
    DltValToIdx[ d ] = idxUpper
}

```

I.8.3.4 Derivation process for the alternative target reference index for temporal motion vector prediction in merge mode

This process is invoked when the current slice is a P or B slice.

The variables AltRefIdxL0 and AltRefIdxL1 are set equal to -1 and the following applies for X in the range of 0 to (slice_type == B) ? 1 : 0, inclusive:

```

zeroIdxLtFlag = RefPicListX[ 0 ] is a short-term reference picture ? 0 : 1
for( i = 1; i <= num_ref_idx_LX_active_minus1 && AltRefIdxLX == -1; i++ )
    if( ( zeroIdxLtFlag && RefPicListX[ i ] is a short-term reference picture ) ||
        ( !zeroIdxLtFlag && RefPicListX[ i ] is a long-term reference picture ) )
        AltRefIdxLX = i

```

(I-57)

I.8.3.5 Derivation process for the target reference index for residual prediction

This process is invoked when the current slice is a P or B slice.

For X in the range of 0 to 1, inclusive, the following applies:

- The variable RpRefIdxLX is set equal to -1 and the variable RpRefPicAvailFlagLX is set equal to 0.
- For i in the range of 0 to NumViews - 1, inclusive, the following applies:
 - The variable RefRpRefAvailFlagLX[ViewOidxList[i]] is set equal to 0.

For X in the range of 0 to (slice_type == B) ? 1 : 0, inclusive, the following applies:

- The variable minPocDiff is set equal to $2^{15} - 1$.
- For i in the range of 0 to num_ref_idx_LX_active_minus1, inclusive, the following applies:
 - The variable pocDiff is set equal to Abs(DiffPicOrderCnt(CurrPic, RefPicListX[i])).
 - When pocDiff is not equal to 0 and pocDiff is less than minPocDiff, the following applies:

minPocDiff = pocDiff (I-58)

RpRefIdxLX = i (I-59)

RpRefPicAvailFlagLX = 1 (I-60)

- When DispAvailFlag is equal to 1 and RpRefPicAvailFlagLX is equal to 1, the following applies for i in the range of 0 to NumActiveRefLayerPics - 1, inclusive:
 - Let picV the picture in the current AU with nuh_layer_id equal to RefPicLayerId[i].
 - When there is a picture picA in one of the reference picture sets RefPicSetLtCurr, RefPicSetStCurrBefore, and RefPicSetStCurrAfter of picV with DiffPicOrderCnt(picA, RefPicListX[RpRefIdxLX]) equal to 0, RefRpRefAvailFlagLX[ViewIdx(RefPicLayerId[i])] is set equal to 1.

The variable RpRefPicAvailFlag is derived as follows:

RpRefPicAvailFlag = (RpRefPicAvailFlagL0 || RpRefPicAvailFlagL1) && DispAvailFlag (I-61)

When RpRefPicAvailFlag is equal to 1 and RefRpRefAvailFlagL0[ViewOidxList[i]] is equal to 1 for any i in the range of 0 to NumViews - 1, inclusive, it is a requirement of bitstream conformance that PicOrderCnt(RefPicList0[RpRefIdxL0]) shall be the same for all slices of a coded picture.

When RpRefPicAvailFlag is equal to 1 and RefRpRefAvailFlagL1[ViewOidxList[i]] is equal to 1 for any i in the range of 0 to NumViews - 1, inclusive, it is a requirement of bitstream conformance that PicOrderCnt(RefPicList1[RpRefIdxL1]) shall be the same for all slices of a coded picture.

I.8.4 Decoding process for coding units coded in intra prediction mode

I.8.4.1 General decoding process for coding units coded in intra prediction mode

The specifications in clause 8.4.1 apply with the following modification:

- All invocations of the process specified in clause 8.4.2 are replaced with invocations of the process specified in

clause I.8.4.2.

- All invocations of the process specified in clause 8.4.4.1 are replaced with invocations of the process specified in clause I.8.4.4.1.

I.8.4.2 Derivation process for luma intra prediction mode

Input to this process is a luma location (xPb, yPb) specifying the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture.

In this process, the luma intra prediction mode $\text{IntraPredModeY}[xPb][yPb]$ is derived.

Table I.2 specifies the value for the intra prediction mode and the associated names.

Table I.2 – Specification of intra prediction mode and associated names

Intra prediction mode	Associated name
0	INTRA_PLANAR
1	INTRA_DC
2..34	INTRA_ANGULAR2..INTRA_ANGULAR34
35	INTRA_WEDGE
36	INTRA_CONTOUR
37	INTRA_SINGLE

$\text{IntraPredModeY}[xPb][yPb]$ labelled 0..34 represents directions of predictions as illustrated in Figure 8-1.

- If $\text{skip_intra_flag}[xPb][yPb]$ is equal to 1, the following applies:
 - If $\text{skip_intra_mode_idx}[xPb][yPb]$ is equal to 0, $\text{IntraPredModeY}[xPb][yPb]$ is set equal to INTRA_ANGULAR26.
 - Otherwise, if $\text{skip_intra_mode_idx}[xPb][yPb]$ is equal to 1, $\text{IntraPredModeY}[xPb][yPb]$ is set equal to INTRA_ANGULAR10.
 - Otherwise ($\text{skip_intra_mode_idx}[xPb][yPb]$ is equal to 2 or 3), $\text{IntraPredModeY}[xPb][yPb]$ is set equal to INTRA_SINGLE.
- Otherwise, if $\text{DimFlag}[xPb][yPb]$ is equal to 1, the following applies:
 - If $\text{depth_intra_mode_idx_flag}[xPb][yPb]$ is equal to 0, $\text{IntraPredModeY}[xPb][yPb]$ is set equal to INTRA_WEDGE.
 - Otherwise ($\text{depth_intra_mode_idx_flag}[xPb][yPb]$ is equal to 1), $\text{IntraPredModeY}[xPb][yPb]$ is set equal to INTRA_CONTOUR.
- Otherwise ($\text{skip_intra_flag}[xPb][yPb]$ and $\text{DimFlag}[xPb][yPb]$ are both equal to 0), $\text{IntraPredModeY}[xPb][yPb]$ is derived by the following ordered steps:
 1. The neighbouring locations (xNbA, yNbA) and (xNbB, yNbB) are set equal to (xPb – 1, yPb) and (xPb, yPb – 1), respectively.
 2. For X being replaced by either A or B, the variables $\text{candIntraPredModeX}$ are derived as follows:
 - The availability derivation process for a block in z-scan order as specified in clause 6.4.1 is invoked with the location (xCurr, yCurr) set equal to (xPb, yPb) and the neighbouring location (xNbY, yNbY) set equal to (xNbX, yNbX) as inputs, and the output is assigned to availableX .
 - The candidate intra prediction mode $\text{candIntraPredModeX}$ is derived as follows:
 - If availableX is equal to FALSE, $\text{candIntraPredModeX}$ is set equal to INTRA_DC.
 - Otherwise, if $\text{CuPredMode}[xNbX][yNbX]$ is not equal to MODE_INTRA or $\text{pcm_flag}[xNbX][yNbX]$ is equal to 1, $\text{candIntraPredModeX}$ is set equal to INTRA_DC,
 - Otherwise, if X is equal to B and $yPb - 1$ is less than ((yPb >> CtbLog2SizeY) << CtbLog2SizeY), $\text{candIntraPredModeB}$ is set equal to INTRA_DC.
 - Otherwise, if $\text{IntraPredModeY}[xNbX][yNbX]$ is greater than 34, $\text{candIntraPredModeX}$ is set equal to INTRA_DC.

- Otherwise, $\text{candIntraPredModeX}$ is set equal to $\text{IntraPredModeY}[xNbX][yNbX]$.
3. The $\text{candModeList}[x]$ with $x = 0..2$ is derived as follows:
- If $\text{candIntraPredModeB}$ is equal to $\text{candIntraPredModeA}$, the following applies:
 - If $\text{candIntraPredModeA}$ is less than 2 (i.e., equal to INTRA_PLANAR or INTRA_DC), $\text{candModeList}[x]$ with $x = 0..2$ is derived as follows:

$$\text{candModeList}[0] = \text{INTRA_PLANAR} \quad (\text{I-62})$$

$$\text{candModeList}[1] = \text{INTRA_DC} \quad (\text{I-63})$$

$$\text{candModeList}[2] = \text{INTRA_ANGULAR26} \quad (\text{I-64})$$
 - Otherwise, $\text{candModeList}[x]$ with $x = 0..2$ is derived as follows:

$$\text{candModeList}[0] = \text{candIntraPredModeA} \quad (\text{I-65})$$

$$\text{candModeList}[1] = 2 + ((\text{candIntraPredModeA} + 29) \% 32) \quad (\text{I-66})$$

$$\text{candModeList}[2] = 2 + ((\text{candIntraPredModeA} - 2 + 1) \% 32) \quad (\text{I-67})$$
 - Otherwise ($\text{candIntraPredModeB}$ is not equal to $\text{candIntraPredModeA}$), the following applies:
 - $\text{candModeList}[0]$ and $\text{candModeList}[1]$ are derived as follows:

$$\text{candModeList}[0] = \text{candIntraPredModeA} \quad (\text{I-68})$$

$$\text{candModeList}[1] = \text{candIntraPredModeB} \quad (\text{I-69})$$
 - If neither of $\text{candModeList}[0]$ and $\text{candModeList}[1]$ is equal to INTRA_PLANAR , $\text{candModeList}[2]$ is set equal to INTRA_PLANAR ,
 - Otherwise, if neither of $\text{candModeList}[0]$ and $\text{candModeList}[1]$ is equal to INTRA_DC , $\text{candModeList}[2]$ is set equal to INTRA_DC ,
 - Otherwise, $\text{candModeList}[2]$ is set equal to INTRA_ANGULAR26 .
4. $\text{IntraPredModeY}[xPb][yPb]$ is derived by applying the following procedure:
- If $\text{prev_intra_luma_pred_flag}[xPb][yPb]$ is equal to 1, the $\text{IntraPredModeY}[xPb][yPb]$ is set equal to $\text{candModeList}[mpm_idx]$.
 - Otherwise, $\text{IntraPredModeY}[xPb][yPb]$ is derived by applying the following ordered steps:
 - 1) The array $\text{candModeList}[x]$, $x = 0..2$ is modified as the following ordered steps:
 - i. When $\text{candModeList}[0]$ is greater than $\text{candModeList}[1]$, both values are swapped as follows:

$$(\text{candModeList}[0], \text{candModeList}[1]) = \text{Swap}(\text{candModeList}[0], \text{candModeList}[1]) \quad (\text{I-70})$$
 - ii. When $\text{candModeList}[0]$ is greater than $\text{candModeList}[2]$, both values are swapped as follows:

$$(\text{candModeList}[0], \text{candModeList}[2]) = \text{Swap}(\text{candModeList}[0], \text{candModeList}[2]) \quad (\text{I-71})$$
 - iii. When $\text{candModeList}[1]$ is greater than $\text{candModeList}[2]$, both values are swapped as follows:

$$(\text{candModeList}[1], \text{candModeList}[2]) = \text{Swap}(\text{candModeList}[1], \text{candModeList}[2]) \quad (\text{I-72})$$
 - 2) $\text{IntraPredModeY}[xPb][yPb]$ is derived by the following ordered steps:
 - i. $\text{IntraPredModeY}[xPb][yPb]$ is set equal to $\text{rem_intra_luma_pred_mode}[xPb][yPb]$.
 - ii. For i equal to 0 to 2, inclusive, when $\text{IntraPredModeY}[xPb][yPb]$ is greater than or equal to $\text{candModeList}[i]$, the value of $\text{IntraPredModeY}[xPb][yPb]$ is incremented by one.

I.8.4.3 Derivation process for chroma intra prediction mode

The specifications in clause 8.4.3 apply.

I.8.4.4 Decoding process for intra blocks

I.8.4.4.1 General decoding process for intra blocks

Inputs to this process are:

- a sample location (xTb0, yTb0) specifying the top-left sample of the current transform block relative to the top-left sample of the current picture,
- a variable log2TrafoSize specifying the size of the current transform block,
- a variable trafoDepth specifying the hierarchy depth of the current block relative to the coding unit,
- a variable predModeIntra specifying the intra prediction mode,
- a variable cIdx specifying the colour component of the current block.

Output of this process is a modified reconstructed picture before deblocking filtering.

The luma sample location (xTbY, yTbY) specifying the top-left sample of the current luma transform block relative to the top-left luma sample of the current picture is derived as follows:

$$(xTbY, yTbY) = (cIdx == 0) ? (xTb0, yTb0) : (xTb0 * SubWidthC, yTb0 * SubHeightC) \quad (I-73)$$

The variable splitFlag is derived as follows:

- If DcOnlyFlag[xTbY][yTbY] is equal to 1, the following applies:

$$splitFlag = !DimFlag[xTbY][yTbY] \&\& (log2TrafoSize > MaxTbLog2SizeY) \quad (I-74)$$

- Otherwise, if skip_intra_flag[xTbY][yTbY] is equal to 1, splitFlag is set equal to 0.
- Otherwise, if cIdx is equal to 0, splitFlag is set equal to split_transform_flag[xTbY][yTbY][trafoDepth].
- Otherwise, if all of the following conditions are true, splitFlag is set equal to 1.
 - cIdx is greater than 0
 - split_transform_flag[xTbY][yTbY][trafoDepth] is equal to 1
 - log2TrafoSize is greater than 2
- Otherwise, splitFlag is set equal to 0.

Depending on the value of splitFlag, the following applies:

- If splitFlag is equal to 1, the following ordered steps apply:

1. The variables xTb1 and yTb1 are derived as follows:

- If cIdx is equal to 0 or ChromaArrayType is not equal to 2, the following applies:
 - The variable xTb1 is set equal to $xTb0 + (1 \ll (log2TrafoSize - 1))$.
 - The variable yTb1 is set equal to $yTb0 + (1 \ll (log2TrafoSize - 1))$.
- Otherwise (ChromaArrayType is equal to 2 and cIdx is greater than 0), the following applies:
 - The variable xTb1 is set equal to $xTb0 + (1 \ll (log2TrafoSize - 1))$.
 - The variable yTb1 is set equal to $yTb0 + (2 \ll (log2TrafoSize - 1))$.

2. The general decoding process for intra blocks as specified in this clause is invoked with the location (xTb0, yTb0), the variable log2TrafoSize set equal to $log2TrafoSize - 1$, the variable trafoDepth set equal to $trafoDepth + 1$, the intra prediction mode predModeIntra, and the variable cIdx as inputs, and the output is a modified reconstructed picture before deblocking filtering.
3. The general decoding process for intra blocks as specified in this clause is invoked with the location (xTb1, yTb0), the variable log2TrafoSize set equal to $log2TrafoSize - 1$, the variable trafoDepth set equal to $trafoDepth + 1$, the intra prediction mode predModeIntra, and the variable cIdx as inputs, and the output is a modified reconstructed picture before deblocking filtering.
4. The general decoding process for intra blocks as specified in this clause is invoked with the location (xTb0, yTb1), the variable log2TrafoSize set equal to $log2TrafoSize - 1$, the variable trafoDepth set equal to $trafoDepth + 1$, the intra prediction mode predModeIntra, and the variable cIdx as inputs, and the output is a modified reconstructed picture before deblocking filtering.
5. The general decoding process for intra blocks as specified in this clause is invoked with the location (xTb1, yTb1), the variable log2TrafoSize set equal to $log2TrafoSize - 1$, the variable trafoDepth set equal to $trafoDepth + 1$, the intra prediction mode predModeIntra, and the variable cIdx as inputs, and the output is a modified reconstructed picture before deblocking filtering.

- Otherwise (splitFlag is equal to 0), for the variable blkIdx proceeding over the values 0..(cIdx > 0 && ChromaArrayType == 2 ? 1 : 0), the following ordered steps apply:
 1. The variable nTbS is set equal to $1 \ll \log_2 \text{TrafoSize}$.
 2. The variable yTbOffset is set equal to $\text{blkIdx} * \text{nTbS}$.
 3. The variable yTbOffsetY is set equal to $\text{yTbOffset} * \text{SubHeightC}$.
 4. The variable residualDpcm is derived as follows:
 - If all of the following conditions are true, residualDpcm is set equal to 1.
 - implicit_rdpem_enabled_flag is equal to 1.
 - either $\text{transform_skip_flag}[\text{xTbY}][\text{yTbY} + \text{yTbOffsetY}][\text{cIdx}]$ is equal to 1, or $\text{cu_transquant_bypass_flag}$ is equal to 1.
 - either predModeIntra is equal to 10, or predModeIntra is equal to 26.
 - Otherwise, residualDpcm is set equal to 0.
 5. The general intra sample prediction process as specified in clause I.8.4.4.2.1 is invoked with the transform block location (xTb0, yTb0 + yTbOffset), the intra prediction mode predModeIntra , the transform block size nTbS, and the variable cIdx as inputs, and the output is an (nTbS)x(nTbS) array predSamples.
 6. The variable residualFlag is set equal to $!(\text{skip_intra_flag}[\text{xTb0}][\text{xTb0}] \mid \mid \text{DcOnlyFlag}[\text{xTb0}][\text{xTb0}])$ and depending on the value of residualFlag, the following applies:
 - If residualFlag is equal to 1, the following applies:
 - The scaling and transformation process as specified in clause 8.6.2 is invoked with the luma location (xTbY, yTbY + yTbOffsetY), the variable trafoDepth, the variable cIdx, and the transform size trafoSize set equal to nTbS as inputs, and the output is an (nTbS)x(nTbS) array resSamples.
 - When residualDpcm is equal to 1, the directional residual modification process for blocks using a transform bypass as specified in clause 8.6.5 is invoked with the variable mDir set equal to $\text{predModeIntra} / 26$, the variable nTbS, and the (nTbS)x(nTbS) array r set equal to the array resSamples as inputs, and the output is a modified (nTbS)x(nTbS) array resSamples.
 - When cross_component_prediction_enabled_flag is equal to 1, ChromaArrayType is equal to 3, and cIdx is not equal to 0, the residual modification process for transform blocks using cross-component prediction as specified in clause 8.6.6 is invoked with the current luma transform block location (xTbY, yTbY), the variable nTbS, the variable cIdx, the (nTbS)x(nTbS) array r_Y set equal to the corresponding luma residual sample array resSamples of the current transform block, and the (nTbS)x(nTbS) array r set equal to the array resSamples as inputs, and the output is a modified (nTbS)x(nTbS) array resSamples.
 - Otherwise (residualFlag is equal to 0), for $x, y = 0..nTbS - 1$, $\text{resSamples}[x][y]$ is set equal to 0.
 7. The picture construction process prior to in-loop filtering for a colour component as specified in clause 8.6.7 is invoked with the transform block location (xTb0, yTb0 + yTbOffset), the variables nCurrSw and nCurrSh both set equal to nTbS, the variable cIdx, the (nTbS)x(nTbS) array predSamples, and the (nTbS)x(nTbS) array resSamples as inputs.

When trafoDepth is equal to 0, $\text{DcOnlyFlag}[\text{xTb0}][\text{yTb0}]$ is equal to 1, and $\text{DimFlag}[\text{xTb0}][\text{yTb0}]$ is equal to 0, the depth DC offset assignment process as specified in clause I.8.4.4.3 is invoked with the location (xTb0, yTb0), and the transform size trafoSize set equal to nTbS as inputs.

I.8.4.4.2 Intra sample prediction

I.8.4.4.2.1 General intra sample prediction

Inputs to this process are:

- a sample location (xTbCmp, yTbCmp) specifying the top-left sample of the current transform block relative to the top-left sample of the current picture,
- a variable predModeIntra specifying the intra prediction mode,
- a variable nTbS specifying the transform block size,
- a variable cIdx specifying the colour component of the current block.

Outputs of this process are the predicted samples $\text{predSamples}[x][y]$, with $x, y = 0..nTbS - 1$.

The $nTbS * 4 + 1$ neighbouring samples $p[x][y]$ that are constructed samples prior to the deblocking filter process, with $x = -1, y = -1..nTbS * 2 - 1$ and $x = 0..nTbS * 2 - 1, y = -1$, are derived as follows:

- The neighbouring location $(xNbCmp, yNbCmp)$ is specified by:

$$(xNbCmp, yNbCmp) = (xTbCmp + x, yTbCmp + y) \quad (I-75)$$

- The current luma location $(xTbY, yTbY)$ and the neighbouring luma location $(xNbY, yNbY)$ are derived as follows:

$$(xTbY, yTbY) = (cIdx == 0) ? (xTbCmp, yTbCmp) : (xTbCmp * SubWidthC, yTbCmp * SubHeightC) \quad (I-76)$$

$$(xNbY, yNbY) = (cIdx == 0) ? (xNbCmp, yNbCmp) : (xNbCmp * SubWidthC, yNbCmp * SubHeightC) \quad (I-77)$$

- The availability derivation process for a block in z-scan order as specified in clause 6.4.1 is invoked with the current luma location $(xCurr, yCurr)$ set equal to $(xTbY, yTbY)$ and the neighbouring luma location $(xNbY, yNbY)$ as inputs, and the output is assigned to `availableN`.
- Each sample $p[x][y]$ is derived as follows:
 - If one or more of the following conditions are true, the sample $p[x][y]$ is marked as "not available for intra prediction":
 - The variable `availableN` is equal to `FALSE`.
 - `CuPredMode[xNbY][yNbY]` is not equal to `MODE_INTRA` and `constrained_intra_pred_flag` is equal to 1.
 - Otherwise, the sample $p[x][y]$ is marked as "available for intra prediction" and the sample at the location $(xNbCmp, yNbCmp)$ is assigned to $p[x][y]$.

When at least one sample $p[x][y]$ with $x = -1, y = -1..nTbS * 2 - 1$ and $x = 0..nTbS * 2 - 1, y = -1$ is marked as "not available for intra prediction" and `predModeIntra` is not equal to `INTRA_SINGLE`, the reference sample substitution process for intra sample prediction in clause 8.4.4.2.2 is invoked with the samples $p[x][y]$ with $x = -1, y = -1..nTbS * 2 - 1$ and $x = 0..nTbS * 2 - 1, y = -1, nTbS$, and `cIdx` as inputs, and the modified samples $p[x][y]$ with $x = -1, y = -1..nTbS * 2 - 1$ and $x = 0..nTbS * 2 - 1, y = -1$ as output.

Depending on the value of `predModeIntra`, the following ordered steps apply:

1. When `predModeIntra` is in the range of 0 to 34, inclusive, `intra_smoothing_disabled_flag` is equal to 0, and either `cIdx` is equal to 0 or `ChromaArrayType` is equal to 3, the filtering process of neighbouring samples specified in clause 8.4.4.2.3 is invoked with the sample array `p`, the transform block size `nTbS` and the colour component index `cIdx` as inputs, and the output is reassigned to the sample array `p`.
2. The intra sample prediction process according to `predModeIntra` applies as follows:
 - If `predModeIntra` is equal to `INTRA_PLANAR`, the corresponding intra prediction mode specified in clause 8.4.4.2.4 is invoked with the sample array `p` and the transform block size `nTbS` as inputs, and the output is the predicted sample array `predSamples`.
 - Otherwise, if `predModeIntra` is equal to `INTRA_DC`, the corresponding intra prediction mode specified in clause 8.4.4.2.5 is invoked with the sample array `p`, the transform block size `nTbS`, and the colour component index `cIdx` as inputs, and the output is the predicted sample array `predSamples`.
 - Otherwise, if `predModeIntra` is in the range of `INTRA_ANGULAR2..INTRA_ANGULAR34`, the corresponding intra prediction mode specified in clause 8.4.4.2.6 is invoked with the intra prediction mode `predModeIntra`, the sample array `p`, the transform block size `nTbS`, and the colour component index `cIdx` as inputs, and the output is the predicted sample array `predSamples`.
 - Otherwise, if `predModeIntra` is equal to `INTRA_WEDGE`, the corresponding intra prediction mode specified in clause 8.4.4.2.2 is invoked with the location $(xTbY, yTbY)$, the sample array `p`, and the transform block size `nTbS` as inputs, and the output is the predicted sample array `predSamples`.
 - Otherwise, if `predModeIntra` is equal to `INTRA_CONTOUR`, the corresponding intra prediction mode specified in clause 8.4.4.2.3 is invoked with the location $(xTbY, yTbY)$, the sample array `p`, and the transform block size `nTbS` as inputs, and the output is the predicted sample array `predSamples`.
 - Otherwise, if `predModeIntra` is equal to `INTRA_SINGLE`, the corresponding intra prediction mode specified in clause 8.4.4.2.4 is invoked with the location $(xTbY, yTbY)$, the sample array `p`, and the transform block size `nTbS` as inputs, and the output is the predicted sample array `predSamples`.

I.8.4.4.2.2 Specification of intra prediction mode INTRA_WEDGE

Inputs to this process are:

- a luma location (xTb , yTb) specifying the top-left sample of the current transform block relative to the top-left luma sample of the current picture,
- the neighbouring samples $p[x][y]$, with $x = -1, y = -1..nTbS * 2 - 1$ and $x = 0..nTbS * 2 - 1, y = -1$,
- a variable $nTbS$ specifying the transform block size.

Output of this process are the predicted samples $predSamples[x][y]$, with $x, y = 0..nTbS - 1$.

The list `WedgePatternTable` and the variable `NumWedgePattern` are specified by the derivation process for a wedgelet partition pattern table in clause I.6.6.

The partition pattern $wedgePattern[x][y]$ with $x, y = 0..nTbS - 1$ is derived as follows:

$$wedgePattern = WedgePatternTable[\text{Log2}(nTbS)][wedge_full_tab_idx[xTb][yTb]] \quad (I-78)$$

The depth sub-block partition DC value derivation and assignment process as specified in clause I.8.4.4.2.5 is invoked with the neighbouring samples $p[x][y]$, the partition pattern $wedgePattern[x][y]$, the luma location (xTb , yTb), and the transform size $nTbS$ as inputs, and the output is assigned to $predSamples[x][y]$.

I.8.4.4.2.3 Specification of intra prediction mode INTRA_CONTOUR

Inputs to this process are:

- a luma location (xTb , yTb) specifying the top-left sample of the current transform block relative to the top-left luma sample of the current picture,
- the neighbouring samples $p[x][y]$, with $x = -1, y = -1..nTbS * 2 - 1$ and $x = 0..nTbS * 2 - 1, y = -1$,
- a variable $nTbS$ specifying the transform block size.

Output of this process are the predicted samples $predSamples[x][y]$, with $x, y = 0..nTbS - 1$.

The partition pattern $contourPattern[x][y]$, with $x, y = 0..nTbS - 1$ is derived as follows:

- The variable `refSamples` is set equal to the reconstructed picture sample array S_L of the picture `TexturePic`.
- The variable `threshVal` is derived as follows:

$$\begin{aligned} threshVal = & (refSamples[xTb][yTb] + refSamples[xTb][yTb + nTbS - 1] \\ & + refSamples[xTb + nTbS - 1][yTb] + refSamples[xTb + nTbS - 1][yTb + nTbS - 1]) \gg 2 \end{aligned} \quad (I-79)$$

- For $x = 0..nTbS - 1$ and $y = 0..nTbS - 1$, the following applies:

$$contourPattern[x][y] = (refSamples[xTb + x][yTb + y] > threshVal) \quad (I-80)$$

The depth sub-block partition DC value derivation and assignment process as specified in clause I.8.4.4.2.5 is invoked with the neighbouring samples $p[x][y]$, the partition pattern $contourPattern[x][y]$, the luma location (xTb , yTb), and the transform size $nTbS$ as inputs, and the output is assigned to $predSamples[x][y]$.

I.8.4.4.2.4 Specification of intra prediction mode INTRA_SINGLE

Inputs to this process are:

- a luma location (xTb , yTb) specifying the top-left sample of the current transform block relative to the top-left luma sample of the current picture,
- the neighbouring samples $p[x][y]$, with $x = -1, y = -1..nTbS * 2 - 1$ and $x = 0..nTbS * 2 - 1, y = -1$,
- a variable $nTbS$ specifying the transform block size.

Output of this process are the predicted samples $predSamples[x][y]$, with $x, y = 0..nTbS - 1$.

Depending on the value of $skip_intra_mode_idx[xTb][yTb]$, the luma location (xN , yN) is derived as follows:

- If $skip_intra_mode_idx[xTb][yTb]$ is equal to 2, (xN , yN) is set equal to ($-1, nTbS \gg 1$).
- Otherwise ($skip_intra_mode_idx[xTb][yTb]$ is equal to 3), (xN , yN) is set equal to ($nTbS \gg 1, -1$).

The variable `singleSampleVal` is derived as follows:

- If $p[xN][yN]$ is marked as "available for intra prediction", `singleSampleVal` is set equal to $p[xN][yN]$.

- Otherwise ($p[x][y]$ is marked as "not available for intra prediction"), $singleSampleVal$ is set equal to $(1 \ll (\text{BitDepth}_Y - 1))$

The values of the prediction samples $predSamples[x][y]$, with $x, y = 0..nTbS - 1$, are derived as follows:

$$predSamples[x][y] = singleSampleVal \quad (I-81)$$

I.8.4.4.2.5 Depth sub-block partition DC value derivation and assignment process

Inputs to this process are:

- the neighbouring samples $p[x][y]$, with $x = -1, y = -1..nTbS * 2 - 1$ and $x = 0..nTbS * 2 - 1, y = -1$,
- a partition pattern $partitionPattern[x][y]$, with $x, y = 0..nTbS - 1$, specifying the sub-block partitions of the current prediction block,
- a luma location (xTb, yTb) specifying the top-left sample of the current transform block relative to the top-left luma sample of the current picture,
- a variable $nTbS$ specifying the transform block size.

Output of this process are the predicted samples $predSamples[x][y]$, with $x, y = 0..nTbS - 1$.

The variables $vertEdgeFlag$ and $horEdgeFlag$ are derived as follows:

$$vertEdgeFlag = (partitionPattern[0][0] \neq partitionPattern[nTbS - 1][0]) \quad (I-82)$$

$$horEdgeFlag = (partitionPattern[0][0] \neq partitionPattern[0][nTbS - 1]) \quad (I-83)$$

The variables $dcValBR$ and $dcValLT$ are derived as follows:

- If $vertEdgeFlag$ is equal to $horEdgeFlag$, the following applies:

- The variable $dcValBR$ is derived as follows:

- If $horEdgeFlag$ is equal to 1, the following applies:

$$dcValBR = ((p[-1][nTbS - 1] + p[nTbS - 1][-1]) \gg 1) \quad (I-84)$$

- Otherwise ($horEdgeFlag$ is equal to 0), the following applies:

$$vertAbsDiff = \text{Abs}(p[-1][0] - p[-1][nTbS * 2 - 1]) \quad (I-85)$$

$$horAbsDiff = \text{Abs}(p[0][-1] - p[nTbS * 2 - 1][-1]) \quad (I-86)$$

$$dcValBR = (horAbsDiff > vertAbsDiff) ? p[nTbS * 2 - 1][-1] : p[-1][nTbS * 2 - 1] \quad (I-87)$$

- The variable $dcValLT$ is derived as follows:

$$dcValLT = (p[-1][0] + p[0][-1]) \gg 1 \quad (I-88)$$

- Otherwise ($horEdgeFlag$ is not equal to $vertEdgeFlag$), the following applies:

$$dcValBR = horEdgeFlag ? p[-1][nTbS - 1] : p[nTbS - 1][-1] \quad (I-89)$$

$$dcValLT = horEdgeFlag ? p[(nTbS - 1) \gg 1][-1] : p[-1][(nTbS - 1) \gg 1] \quad (I-90)$$

The predicted sample values $predSamples[x][y]$ are derived as follows:

- For x in the range of 0 to $(nTbS - 1)$, inclusive, the following applies:
 - For y in the range of 0 to $(nTbS - 1)$, inclusive, the following applies:

- The variables $predDcVal$ and $dcOffset$ are derived as follows:

$$predDcVal = (partitionPattern[x][y] == partitionPattern[0][0]) ? dcValLT : dcValBR \quad (I-91)$$

$$dcOffset = DcOffset[xTb][yTb][partitionPattern[x][y]] \quad (I-92)$$

- If $DltFlag[nuh_layer_id]$ is equal to 0, the following applies:

$$predSamples[x][y] = predDcVal + dcOffset \quad (I-93)$$

- Otherwise ($DltFlag[nuh_layer_id]$ is equal to 1), the following applies:

$$predSamples[x][y] = DltIdxToVal[Clip1_Y(DltValToIdx[predDcVal] + dcOffset)] \quad (I-94)$$

I.8.4.4.3 Depth DC offset assignment process

Inputs to this process are:

- a luma location (x_{Tb} , y_{Tb}) specifying the top-left luma sample of the current transform block relative to the top-left luma sample of the current picture,
- a variable n_{TbS} specifying the transform block size.

Output of this process is a modified reconstructed picture S_L before deblocking filtering.

Depending on the value of $DltFlag[nuh_layer_id]$, the variable $dcVal$ is derived as follows:

- If $DltFlag[nuh_layer_id]$ is equal to 0, $dcVal$ is set equal to $DcOffset[x_{Tb}][y_{Tb}][0]$.
- Otherwise ($DltFlag[nuh_layer_id]$ is equal to 1), $dcVal$ is derived as follows:

$$dcPred = (S_L[x_{Tb}][y_{Tb}] + S_L[x_{Tb}][y_{Tb} + n_{TbS} - 1] + S_L[x_{Tb} + n_{TbS} - 1][y_{Tb}] + S_L[x_{Tb} + n_{TbS} - 1][y_{Tb} + n_{TbS} - 1] + 2) >> 2 \quad (I-95)$$

$$dcVal = DltIdxToVal[Clip1_Y(DltValToIdx[dcPred] + DcOffset[x_{Tb}][y_{Tb}][0])] - dcPred \quad (I-96)$$

For $x, y = 0..n_{TbS} - 1$, the variable $S_L[x][y]$ is modified as follows:

$$S_L[x_{Tb} + x][y_{Tb} + y] = Clip1_Y(S_L[x_{Tb} + x][y_{Tb} + y] + dcVal) \quad (I-97)$$

I.8.5 Decoding process for coding units coded in inter prediction mode

I.8.5.1 General decoding process for coding units coded in inter prediction mode

Inputs to this process are:

- a luma location (x_{Cb} , y_{Cb}) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a variable $\log2CbSize$ specifying the size of the current coding block.

Output of this process is a modified reconstructed picture before deblocking filtering.

The derivation process for quantization parameters as specified in clause 8.6.1 is invoked with the luma location (x_{Cb} , y_{Cb}) as input.

The variable $nCbS_L$ is set equal to $1 \ll \log2CbSize$. When $ChromaArrayType$ is not equal to 0, the variable $nCbSw_C$ is set equal to $(1 \ll \log2CbSize) / SubWidthC$ and the variable $nCbSh_C$ is set equal to $(1 \ll \log2CbSize) / SubHeightC$.

The decoding process for coding units coded in inter prediction mode consists of following ordered steps:

1. When $DisparityDerivationFlag$ is equal to 1, the following applies:
 - If $DepthFlag$ is equal to 0, the derivation process for a disparity vector for texture layers as specified in clause I.8.5.5 is invoked with the luma location (x_{Cb} , y_{Cb}) and the coding block size $nCbS_L$ as inputs.
 - Otherwise ($DepthFlag$ is equal to 1), the derivation process for a disparity vector for depth layers as specified in clause I.8.5.6 is invoked with the luma location (x_{Cb} , y_{Cb}) and the coding block size $nCbS_L$ as inputs.
2. The inter prediction process as specified in clause I.8.5.2 is invoked with the luma location (x_{Cb} , y_{Cb}) and the luma coding block size $\log2CbSize$ as inputs, and the output is the array $predSamples_L$ and, when $ChromaArrayType$ is not equal to 0, the arrays $predSamples_{Cb}$, and $predSamples_{Cr}$.
3. The decoding process for the residual signal of coding units coded in inter prediction mode specified in clause I.8.5.4 is invoked with the luma location (x_{Cb} , y_{Cb}) and the luma coding block size $\log2CbSize$ as inputs, and the outputs are the array $resSamples_L$ and, when $ChromaArrayType$ is not equal to 0, the arrays $resSamples_{Cb}$, and $resSamples_{Cr}$.
4. The reconstructed samples of the current coding unit are derived as follows:
 - The picture construction process prior to in-loop filtering for a colour component as specified in clause 8.6.7 is invoked with the luma coding block location (x_{Cb} , y_{Cb}), the variable $nCurrSw$ set equal to $nCbS_L$, the variable $nCurrSh$ set equal to $nCbS_L$, the variable $cIdx$ set equal to 0, the $(nCbS_L) \times (nCbS_L)$ array $predSamples$ set equal to $predSamples_L$, and the $(nCbS_L) \times (nCbS_L)$ array $resSamples$ set equal to $resSamples_L$ as inputs.
 - When $ChromaArrayType$ is not equal to 0, the picture construction process prior to in-loop filtering for a colour component as specified in clause 8.6.7 is invoked with the chroma coding block location ($x_{Cb} / SubWidthC$, $y_{Cb} / SubHeightC$), the variable $nCurrSw$ set equal to $nCbSw_C$, the variable $nCurrSh$

set equal to $nCbSh_C$, the variable $cIdx$ set equal to 1, the $(nCbSw_C) \times (nCbSh_C)$ array $predSamples$ set equal to $predSamples_{Cb}$, and the $(nCbSw_C) \times (nCbSh_C)$ array $resSamples$ set equal to $resSamples_{Cb}$ as inputs.

- When $ChromaArrayType$ is not equal to 0, the picture construction process prior to in-loop filtering for a colour component as specified in clause 8.6.7 is invoked with the chroma coding block location ($xCb / SubWidthC$, $yCb / SubHeightC$), the variable $nCurrSw$ set equal to $nCbSw_C$, the variable $nCurrSh$ set equal to $nCbSh_C$, the variable $cIdx$ set equal to 2, the $(nCbSw_C) \times (nCbSh_C)$ array $predSamples$ set equal to $predSamples_{Cr}$, and the $(nCbSw_C) \times (nCbSh_C)$ array $resSamples$ set equal to $resSamples_{Cr}$ as inputs.

I.8.5.2 Inter prediction process

The specifications in clause 8.5.2 apply with the following modification:

- All invocations of the process specified in clause 8.5.3 are replaced with invocations of the process specified in clause I.8.5.3.

I.8.5.3 Decoding process for prediction units in inter prediction mode

I.8.5.3.1 General

Inputs to this process are:

- a luma location (xCb , yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a luma location (xBl , yBl) specifying the top-left sample of the current luma prediction block relative to the top-left sample of the current luma coding block,
- a variable $nCbS$ specifying the size of the current luma coding block,
- a variable $nPbW$ specifying the width of the current luma prediction block,
- a variable $nPbH$ specifying the height of the current luma prediction block,
- a variable $partIdx$ specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

- an $(nCbS_L) \times (nCbS_L)$ array $predSamples_L$ of luma prediction samples, where $nCbS_L$ is derived as specified below,
- when $ChromaArrayType$ is not equal to 0, an $(nCbSw_C) \times (nCbSh_C)$ array $predSamples_{Cb}$ of chroma prediction samples for the component Cb , where $nCbSw_C$ and $nCbSh_C$ are derived as specified below,
- when $ChromaArrayType$ is not equal to 0, an $(nCbSw_C) \times (nCbSh_C)$ array $predSamples_{Cr}$ of chroma prediction samples for the component Cr , where $nCbSw_C$ and $nCbSh_C$ are derived as specified below.

The variable $nCbS_L$ is set equal to $nCbS$. When $ChromaArrayType$ is not equal to 0, the variable $nCbSw_C$ is set equal to $nCbS / SubWidthC$ and the variable $nCbSh_C$ is set equal to $nCbS / SubHeightC$.

The decoding process for prediction units in inter prediction mode consists of the following ordered steps:

1. The derivation process for motion vector components and reference indices as specified in clause I.8.5.3.2 is invoked with the luma coding block location (xCb , yCb), the luma prediction block location (xBl , yBl), the luma coding block size block $nCbS$, the luma prediction block width $nPbW$, the luma prediction block height $nPbH$, and the prediction unit index $partIdx$ as inputs, and the luma motion vectors $mvL0$ and $mvL1$, when $ChromaArrayType$ is not equal to 0, the chroma motion vectors $mvCL0$ and $mvCL1$, the reference indices $refIdxL0$ and $refIdxL1$, the prediction list utilization flags $predFlagL0$ and $predFlagL1$, and the flag $subPbMotionFlag$ as outputs.
2. Depending on the value of $subPbMotionFlag$ and $DbbpFlag[xCb][yCb]$, the following applies:
 - If both $subPbMotionFlag$ and $DbbpFlag[xCb][yCb]$ are equal to 0, the decoding process for inter sample prediction as specified in clause I.8.5.3.3.1 is invoked with the luma coding block location (xCb , yCb), the luma prediction block location (xBl , yBl), the luma coding block size block $nCbS$, the luma prediction block width $nPbW$, the luma prediction block height $nPbH$, the luma motion vectors $mvL0$ and $mvL1$, when $ChromaArrayType$ is not equal to 0, the chroma motion vectors $mvCL0$ and $mvCL1$, the reference indices $refIdxL0$ and $refIdxL1$, and the prediction list utilization flags $predFlagL0$ and $predFlagL1$ as inputs, and the inter prediction samples ($predSamples$) that are an $(nCbS_L) \times (nCbS_L)$ array $predSamples_L$ of prediction luma samples and, when $ChromaArrayType$ is not equal to 0, two $(nCbSw_C) \times (nCbSh_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$ of prediction chroma samples, one for each of the chroma components Cb and Cr , as outputs.
 - Otherwise, if $subPbMotionFlag$ is equal to 1, the decoding process for inter sample prediction for rectangular

sub-block partitions as specified in clause I.8.5.3.4 is invoked with the luma coding block location (x_{Cb} , y_{Cb}), the luma prediction block location (x_{Bl} , y_{Bl}), the luma coding block size block n_{CbS} , the luma prediction block width n_{PbW} , and the luma prediction block height n_{PbH} as inputs, and the inter prediction samples that are an $(n_{CbS_L}) \times (n_{CbS_L})$ array $predSamples_L$ of prediction luma samples and, when ChromaArrayType is not equal to 0, two $(n_{CbSw_C}) \times (n_{CbSh_C})$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$ of chroma prediction samples, one for each of the chroma components Cb and Cr, as outputs.

- Otherwise ($DbbpFlag[x_{Cb}][y_{Cb}]$ is equal to 1), the decoding process for inter sample prediction for depth predicted sub-block partitions as specified in clause I.8.5.3.5 is invoked with the luma coding block location (x_{Cb} , y_{Cb}), the luma coding block size n_{CbS} , the luma motion vectors $mvL0$ and $mvL1$, when ChromaArrayType is not equal to 0, the chroma motion vectors $mvCL0$ and $mvCL1$, the reference indices $refIdxL0$ and $refIdxL1$, the prediction list utilization flags $predFlagL0$ and $predFlagL1$, and the variable $partIdx$ as inputs, and the inter prediction samples that are an $(n_{CbS_L}) \times (n_{CbS_L})$ array $predSamples_L$ of prediction luma samples and, when ChromaArrayType is not equal to 0, two $(n_{CbSw_C}) \times (n_{CbSh_C})$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$ of chroma prediction samples, one for each of the chroma components Cb and Cr, as outputs.

For use in derivation processes of variables invoked later in the decoding process, the following assignments are made for $x = x_{Bl} \cdot x_{Bl} + n_{PbW} - 1$ and $y = y_{Bl} \cdot y_{Bl} + n_{PbH} - 1$:

$$MvL0[x_{Cb} + x][y_{Cb} + y] = subPbMotionFlag ? SubPbArrayMvL0[x_{Cb} + x][y_{Cb} + y] : mvL0 \quad (I-98)$$

$$MvL1[x_{Cb} + x][y_{Cb} + y] = subPbMotionFlag ? SubPbArrayMvL1[x_{Cb} + x][y_{Cb} + y] : mvL1 \quad (I-99)$$

$$RefIdxL0[x_{Cb} + x][y_{Cb} + y] = subPbMotionFlag ? SubPbArrayRefIdxL0[x_{Cb} + x][y_{Cb} + y] : refIdxL0 \quad (I-100)$$

$$RefIdxL1[x_{Cb} + x][y_{Cb} + y] = subPbMotionFlag ? SubPbArrayRefIdxL1[x_{Cb} + x][y_{Cb} + y] : refIdxL1 \quad (I-101)$$

$$PredFlagL0[x_{Cb} + x][y_{Cb} + y] = subPbMotionFlag ? SubPbArrayPredFlagL0[x_{Cb} + x][y_{Cb} + y] : predFlagL0 \quad (I-102)$$

$$PredFlagL1[x_{Cb} + x][y_{Cb} + y] = subPbMotionFlag ? SubPbArrayPredFlagL1[x_{Cb} + x][y_{Cb} + y] : predFlagL1 \quad (I-103)$$

I.8.5.3.2 Derivation process for motion vector components and reference indices

I.8.5.3.2.1 General

Inputs to this process are:

- a luma location (x_{Cb} , y_{Cb}) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a luma location (x_{Bl} , y_{Bl}) of the top-left sample of the current luma prediction block relative to the top-left sample of the current luma coding block,
- a variable n_{CbS} specifying the size of the current luma coding block,
- two variables n_{PbW} and n_{PbH} specifying the width and the height of the luma prediction block,
- a variable $partIdx$ specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

- the luma motion vectors $mvL0$ and $mvL1$,
- when ChromaArrayType is not equal to 0, the chroma motion vectors $mvCL0$ and $mvCL1$,
- the reference indices $refIdxL0$ and $refIdxL1$,
- the prediction list utilization flags $predFlagL0$ and $predFlagL1$,
- the flag $subPbMotionFlag$ specifying, whether specifying whether the current PU is coded using sub-block partitions.

Let (x_{Pb} , y_{Pb}) specify the top-left sample location of the current luma prediction block relative to the top-left luma sample of the current picture where $x_{Pb} = x_{Cb} + x_{Bl}$ and $y_{Pb} = y_{Cb} + y_{Bl}$.

Let the variables $currPic$ and LX be the current picture and $RefPicListX$, with X being 0 or 1, of the current picture, respectively.

The function $LongTermRefPic(aPic, aPb, refIdx, LX)$, with X being 0 or 1, is defined as follows:

- If the picture with index refIdx from reference picture list LX of the slice containing prediction block aPb in the picture aPic was marked as "used for long-term reference" at the time when aPic was the current picture, $\text{LongTermRefPic}(\text{aPic}, \text{aPb}, \text{refIdx}, \text{LX})$ is equal to 1.
- Otherwise, $\text{LongTermRefPic}(\text{aPic}, \text{aPb}, \text{refIdx}, \text{LX})$ is equal to 0.

The variables vspMcFlag , ivMcFlag , and subPbMotionFlag are set equal to 0.

For the derivation of the variables mvL0 and mvL1 , refIdxL0 and refIdxL1 , as well as predFlagL0 and predFlagL1 , the following applies:

- If $\text{merge_flag}[\text{xPb}][\text{yPb}]$ is equal to 1, the derivation process for luma motion vectors for merge mode as specified in clause I.8.5.3.2.7 is invoked with the luma location (xCb, yCb) , the luma location (xPb, yPb) , the variables nCbS , nPbW , nPbH , and the partition index partIdx as inputs, and the output being the luma motion vectors mvL0 , mvL1 , the reference indices refIdxL0 , refIdxL1 , the prediction list utilization flags predFlagL0 and predFlagL1 , the flag ivMcFlag , the flag vspMcFlag , and the flag subPbMotionFlag .
- Otherwise, for X being replaced by either 0 or 1 in the variables predFlagLX , mvLX , and refIdxLX , in PRED_LX , and in the syntax elements ref_idx_IX and MvdLX , the following applies:
 1. The variables refIdxLX and predFlagLX are derived as follows:
 - If $\text{inter_pred_idc}[\text{xPb}][\text{yPb}]$ is equal to PRED_LX or PRED_BI ,

$$\text{refIdxLX} = \text{ref_idx_IX}[\text{xPb}][\text{yPb}] \quad (\text{I-104})$$

$$\text{predFlagLX} = 1 \quad (\text{I-105})$$
 - Otherwise, the variables refIdxLX and predFlagLX are specified by:

$$\text{refIdxLX} = -1 \quad (\text{I-106})$$

$$\text{predFlagLX} = 0 \quad (\text{I-107})$$
 2. The variable mvdLX is derived as follows:

$$\text{mvdLX}[0] = \text{MvdLX}[\text{xPb}][\text{yPb}][0] \quad (\text{I-108})$$

$$\text{mvdLX}[1] = \text{MvdLX}[\text{xPb}][\text{yPb}][1] \quad (\text{I-109})$$
 3. When predFlagLX is equal to 1, the derivation process for luma motion vector prediction in clause I.8.5.3.2.3 is invoked with the luma coding block location (xCb, yCb) , the coding block size nCbS , the luma prediction block location (xPb, yPb) , the variables nPbW , nPbH , refIdxLX , and the partition index partIdx as inputs, and the output being mvpLX .
 4. When predFlagLX is equal to 1, the luma motion vector mvLX is derived as follows:

$$\text{uLX}[0] = (\text{mvpLX}[0] + \text{mvdLX}[0] + 2^{16}) \% 2^{16} \quad (\text{I-110})$$

$$\text{mvLX}[0] = (\text{uLX}[0] \geq 2^{15}) ? (\text{uLX}[0] - 2^{16}) : \text{uLX}[0] \quad (\text{I-111})$$

$$\text{uLX}[1] = (\text{mvpLX}[1] + \text{mvdLX}[1] + 2^{16}) \% 2^{16} \quad (\text{I-112})$$

$$\text{mvLX}[1] = (\text{uLX}[1] \geq 2^{15}) ? (\text{uLX}[1] - 2^{16}) : \text{uLX}[1] \quad (\text{I-113})$$

NOTE – The resulting values of $\text{mvLX}[0]$ and $\text{mvLX}[1]$ as specified above will always be in the range of -2^{15} to $2^{15} - 1$, inclusive.

When ChromaArrayType is not equal to 0 and predFlagLX , with X being 0 or 1, is equal to 1, the derivation process for chroma motion vectors in clause 8.5.3.2.10 is invoked with mvLX as input, and the output being mvCLX .

For use in derivation processes of variables invoked later in the decoding process, the following assignments are made for $\text{x} = \text{xPb}..(\text{xPb} + \text{nPbW} - 1)$, $\text{y} = \text{yPb}..(\text{yPb} + \text{nPbH} - 1)$:

$$\text{IvMcFlag}[\text{x}][\text{y}] = \text{ivMcFlag} \quad (\text{I-114})$$

$$\text{VspMcFlag}[\text{x}][\text{y}] = \text{vspMcFlag} \quad (\text{I-115})$$

I.8.5.3.2.2 Derivation process for an initial merge candidate list

The specifications in clause 8.5.3.2.2 apply with the following modifications:

- Steps 9 and 10 are removed.
- "When slice_type is equal to B, the derivation process for combined bi-predictive merging candidates" is replaced with "When slice_type is equal to B and numCurrMergeCand is less than 5, the derivation process for combined bi-predictive merging candidates".

- "derivation process for merging candidates from neighbouring prediction unit partitions in clause 8.5.3.2.3" is replaced with "derivation process for merging candidates from neighbouring prediction unit partitions in clause I.8.5.3.2.3".
- "temporal luma motion vector prediction in clause 8.5.3.2.8 is invoked" is replaced with "temporal luma motion vector prediction in clause I.8.5.3.2.5 is invoked".
- The outputs of the process are replaced with:
 - the modified luma location (x_{Pb} , y_{Pb}) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
 - the variables n_{PbW} and n_{PbH} specifying the modified width and the height of the luma prediction block,
 - the modified variable $partIdx$ specifying the modified index of the current prediction unit within the current coding unit,
 - the original luma location (x_{OrigP} , y_{OrigP}) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
 - the variables $n_{OrigPbW}$ and $n_{OrigPbH}$ specifying the original width and the height of the luma prediction block,
 - the merging candidate list, $mergeCandList$,
 - the luma motion vectors $mvL0N$ and $mvL1N$, with N being replaced by all elements of $mergeCandList$,
 - the reference indices $refIdxL0N$ and $refIdxL1N$, with N being replaced by all elements of $mergeCandList$,
 - the prediction list utilization flags $predFlagL0N$ and $predFlagL1N$, with N being replaced by all elements of $mergeCandList$.

I.8.5.3.2.3 Derivation process for spatial merging candidates

The specifications in clause 8.5.3.2.3 apply with the following modifications and additions:

- The function $differentMotionLoc(xN, yN, xM, yM)$ is specified as follows:
 - If one or more of the following conditions are true, for X in the range of 0 to 1, inclusive, $differentMotionLoc(xN, yN, xM, yM)$ is set equal to 1:
 - $PredFlagLX[xN][yN]$ is not equal to $PredFlagLX[xM][yM]$.
 - $MvLX[xN][yN]$ is not equal to $MvLX[xM][yM]$.
 - $RefIdxLX[xN][yN]$ is not equal to $RefIdxLX[xM][yM]$.
 - Otherwise, $differentMotionLoc(xN, yN, xM, yM)$ is set equal to 0.
- "the prediction units covering the luma locations (x_{NbA_1} , y_{NbA_1}) and (x_{NbB_1} , y_{NbB_1}) have the same motion vectors and the same reference indices" is replaced with " $differentMotionLoc(x_{NbA_1}, y_{NbA_1}, x_{NbB_1}, y_{NbB_1})$ is equal to 0".
- "the prediction units covering the luma locations (x_{NbB_1} , y_{NbB_1}) and (x_{NbB_0} , y_{NbB_0}) have the same motion vectors and the same reference indices" is replaced with " $differentMotionLoc(x_{NbB_1}, y_{NbB_1}, x_{NbB_0}, y_{NbB_0})$ is equal to 0".
- "the prediction units covering the luma locations (x_{NbA_1} , y_{NbA_1}) and (x_{NbA_0} , y_{NbA_0}) have the same motion vectors and the same reference indices" is replaced with " $differentMotionLoc(x_{NbA_1}, y_{NbA_1}, x_{NbA_0}, y_{NbA_0})$ is equal to 0".
- "prediction units covering the luma locations (x_{NbA_1} , y_{NbA_1}) and (x_{NbB_2} , y_{NbB_2}) have the same motion vectors and the same reference indices" is replaced with " $differentMotionLoc(x_{NbA_1}, y_{NbA_1}, x_{NbB_2}, y_{NbB_2})$ is equal to 0".
- "the prediction units covering the luma locations (x_{NbB_1} , y_{NbB_1}) and (x_{NbB_2} , y_{NbB_2}) have the same motion vectors and the same reference indices" is replaced with " $differentMotionLoc(x_{NbB_1}, y_{NbB_1}, x_{NbB_2}, y_{NbB_2})$ is equal to 0".

I.8.5.3.2.4 Derivation process for luma motion vector prediction

The specifications in clause 8.5.3.2.6 apply with the following modifications:

- "temporal luma motion vector prediction in clause 8.5.3.2.8 is invoked" is replaced by "temporal luma motion vector prediction in clause I.8.5.3.2.5 is invoked".

I.8.5.3.2.5 Derivation process for temporal luma motion vector prediction

Inputs to this process are:

- a luma location (xPb, yPb) specifying the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
- two variables nPbW and nPbH specifying the width and the height of the luma prediction block,
- a reference index refIdxLX, with X being 0 or 1.

Outputs of this process are:

- the motion vector prediction mvLXCol,
- the availability flag availableFlagLXCol.

The variable currPb specifies the current luma prediction block at luma location (xPb, yPb).

The variables mvLXCol and availableFlagLXCol are derived as follows:

- If slice_temporal_mvp_enabled_flag is equal to 0, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.
- Otherwise, the following ordered steps apply:

1. The bottom right collocated motion vector is derived as follows:

$$xColBr = xPb + nPbW \quad (I-116)$$

$$yColBr = yPb + nPbH \quad (I-117)$$

- If $yPb \gg CtbLog2SizeY$ is equal to $yColBr \gg CtbLog2SizeY$, $yColBr$ is less than $pic_height_in_luma_samples$ and $xColBr$ is less than $pic_width_in_luma_samples$, the following applies:
 - The luma location (xColPb, yColPb) is set equal to $((xColBr \gg 4) \ll 4, (yColBr \gg 4) \ll 4)$.
 - The variable colPb specifies the luma prediction block covering the modified location given by (xColPb, yColPb) inside the collocated picture specified by ColPic.
 - Depending on merge_flag[xPb][yPb], the following applies:
 - If merge_flag[xPb][yPb] is equal to 0, the derivation process for collocated motion vectors as specified in clause 8.5.3.2.9 is invoked with currPb, colPb, (xColPb, yColPb) and refIdxLX as inputs, and the output is assigned to mvLXCol and availableFlagLXCol.
 - Otherwise (merge_flag[xPb][yPb] is equal to 1), the derivation process for collocated motion vectors for merge mode as specified in clause I.8.5.3.2.6 is invoked with currPb, colPb, (xColPb, yColPb) and refIdxLX as inputs, and the output is assigned to mvLXCol and availableFlagLXCol.
- Otherwise, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.

2. When availableFlagLXCol is equal to 0, the central collocated motion vector is derived as follows:

$$xColCtr = xPb + (nPbW \gg 1) \quad (I-118)$$

$$yColCtr = yPb + (nPbH \gg 1) \quad (I-119)$$

- The luma location (xColPb, yColPb) is set equal to $((xColCtr \gg 4) \ll 4, (yColCtr \gg 4) \ll 4)$.
- The variable colPb specifies the luma prediction block covering the modified location given by (xColPb, yColPb) inside the collocated picture specified by ColPic.
- Depending on merge_flag[xPb][yPb], the following applies:
 - If merge_flag[xPb][yPb] is equal to 0, the derivation process for collocated motion vectors as specified in clause 8.5.3.2.9 is invoked with currPb, colPb, (xColPb, yColPb) and refIdxLX as inputs, and the output is assigned to mvLXCol and availableFlagLXCol.
 - Otherwise (merge_flag[xPb][yPb] is equal to 1), the derivation process for collocated motion vectors for merge mode as specified in clause I.8.5.3.2.6 is invoked with currPb, colPb, (xColPb, yColPb) and refIdxLX as inputs, and the output is assigned to mvLXCol and availableFlagLXCol.

1.8.5.3.2.6 Derivation process for collocated motion vectors for merge mode

Inputs to this process are:

- a variable currPb specifying the current prediction block,
- a variable colPb specifying the collocated prediction block inside the collocated picture specified by ColPic,
- a luma location (xColPb, yColPb) specifying a sample inside the collocated luma prediction block specified by colPb relative to the top-left luma sample of the collocated picture specified by ColPic,
- a reference index refIdxLX, with X being 0 or 1.

Outputs of this process are:

- the motion vector prediction mvLXCol,
- the availability flag availableFlagLXCol.

The arrays predFlagL0Col[x][y], mvL0Col[x][y], and refIdxL0Col[x][y] are set equal to PredFlagL0[x][y], MvL0[x][y], and RefIdxL0[x][y], respectively, of the collocated picture specified by ColPic, and the arrays predFlagL1Col[x][y], mvL1Col[x][y], and refIdxL1Col[x][y] are set equal to PredFlagL1[x][y], MvL1[x][y], and RefIdxL1[x][y], respectively, of the collocated picture specified by ColPic.

The variables mvLXCol and availableFlagLXCol are derived as follows:

- If colPb is coded in an intra prediction mode, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.
- Otherwise, the following applies:
 - The motion vector mvCol, the reference index refIdxCol, and the reference list identifier listCol are derived as follows:
 - If predFlagL0Col[xColPb][yColPb] is equal to 0, mvCol, refIdxCol, and listCol are set equal to mvL1Col[xColPb][yColPb], refIdxL1Col[xColPb][yColPb], and L1, respectively.
 - Otherwise, if predFlagL0Col[xColPb][yColPb] is equal to 1 and predFlagL1Col[xColPb][yColPb] is equal to 0, mvCol, refIdxCol, and listCol are set equal to mvL0Col[xColPb][yColPb], refIdxL0Col[xColPb][yColPb], and L0, respectively.
 - Otherwise (predFlagL0Col[xColPb][yColPb] is equal to 1 and predFlagL1Col[xColPb][yColPb] is equal to 1), the following assignments are made:
 - If NoBackwardPredFlag is equal to 1, mvCol, refIdxCol, and listCol are set equal to mvLXCol[xColPb][yColPb], refIdxLXCol[xColPb][yColPb], and LX, respectively.
 - Otherwise, mvCol, refIdxCol, and listCol are set equal to mvLNCol[xColPb][yColPb], refIdxLNCol[xColPb][yColPb], and LN, respectively, with N being the value of collocated_from_10_flag.
 - The motion vector mvLXCol and availableFlagLXCol are derived as follows:
 - The variables curLtFlag and colLtFlag are derived as follows:

$$\text{curLtFlag} = \text{LongTermRefPic}(\text{CurrPic}, \text{currPb}, \text{refIdxLX}, \text{LX}) \quad (\text{I-120})$$

$$\text{colLtFlag} = \text{LongTermRefPic}(\text{ColPic}, \text{colPb}, \text{refIdxCol}, \text{listCol}) \quad (\text{I-121})$$
 - When curLtFlag is not equal to colLtFlag and AltRefIdxLX is not equal to -1, the variables AltRefFlagLX, refIdxLX, and curLtFlag are modified as follows:

$$\text{AltRefFlagLX} = 1 \quad (\text{I-122})$$

$$\text{refIdxLX} = \text{AltRefIdxLX} \quad (\text{I-123})$$

$$\text{curLtFlag} = \text{LongTermRefPic}(\text{CurrPic}, \text{currPb}, \text{refIdxLX}, \text{LX}) \quad (\text{I-124})$$
 - The motion vector mvLXCol is modified as follows:
 - If curLtFlag is not equal to colLtFlag, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.
 - Otherwise, the variable availableFlagLXCol is set equal to 1, refPicListCol[refIdxCol] is set to be the picture with reference index refIdxCol in the reference picture list listCol of the slice containing

prediction block colPb in the collocated picture specified by ColPic, and the following applies:

- The variables colDiff and currDiff are set equal to 0 and modified as follows:
 - If curLtFlag is equal to 0, the following applies:

$$\text{colDiff} = \text{DiffPicOrderCnt}(\text{ColPic}, \text{refPicListCol}[\text{refIdxCol}]) \quad (\text{I-125})$$

$$\text{currDiff} = \text{DiffPicOrderCnt}(\text{CurrPic}, \text{RefPicListX}[\text{refIdxLX}]) \quad (\text{I-126})$$
 - Otherwise (curLtFlag is equal to 1), when IvMvScalEnabledFlag is equal to 1, the following applies:

$$\text{colDiff} = \text{ViewIdVal}(\text{ColPic}) - \text{ViewIdVal}(\text{refPicListCol}[\text{refIdxCol}]) \quad (\text{I-127})$$

$$\text{currDiff} = \text{ViewIdVal}(\text{CurrPic}) - \text{ViewIdVal}(\text{RefPicListX}[\text{refIdxLX}]) \quad (\text{I-128})$$
- Depending on the values of colDiff and currDiff, the following applies:
 - If colDiff is equal to currDiff, or colDiff is equal to 0, or currDiff is equal to 0, mvLXCol is derived as follows:

$$\text{mvLXCol} = \text{mvCol} \quad (\text{I-129})$$
 - Otherwise, mvLXCol is derived as a scaled version of the motion vector mvCol as follows:

$$\text{tx} = (16384 + (\text{Abs}(\text{td}) \gg 1)) / \text{td} \quad (\text{I-130})$$

$$\text{distScaleFactor} = \text{Clip3}(-4096, 4095, (\text{tb} * \text{tx} + 32) \gg 6) \quad (\text{I-131})$$

$$\text{mvLXCol} = \text{Clip3}(-32768, 32767, \text{Sign}(\text{distScaleFactor} * \text{mvCol}) * ((\text{Abs}(\text{distScaleFactor} * \text{mvCol}) + 127) \gg 8)) \quad (\text{I-132})$$

where td and tb are derived as follows:

$$\text{td} = \text{Clip3}(-128, 127, \text{colDiff}) \quad (\text{I-133})$$

$$\text{tb} = \text{Clip3}(-128, 127, \text{currDiff}) \quad (\text{I-134})$$

I.8.5.3.2.7 Derivation process for luma motion vectors for merge mode

This process is only invoked when merge_flag[xPb][yPb] is equal to 1, where (xPb, yPb) specifies the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture.

Inputs to this process are:

- a luma location (xCb, yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a luma location (xPb, yPb) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
- a variable nCbS specifying the size of the current luma coding block,
- two variables nPbW and nPbH specifying the width and the height of the luma prediction block,
- a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

- the luma motion vectors mvL0 and mvL1,
- the reference indices refIdxL0 and refIdxL1,
- the prediction list utilization flags predFlagL0 and predFlagL1,
- the flag ivMcFlag specifying whether the current PU is coded using an inter-view predicted merge candidate,
- the flag vspMcFlag specifying whether the current PU is coded using the view synthesis prediction merge candidate,
- the flag subPbMotionFlag specifying whether the current PU is coded using sub-block partitions.

The function differentMotion(N, M) is specified as follows:

- If one or more of the following conditions are true, for X in the range of 0 to 1, inclusive, differentMotion(N, M) is set equal to 1:

- predFlagLXN is not equal to predFlagLXM.
- mvLXN is not equal to mvLXM.
- refIdxLXN is not equal to refIdxLXM.
- Otherwise, differentMotion(N, M) is set equal to 0.

The variables AltRefFlagL0 and AltRefFlagL1 are set equal to 0.

The motion vectors mvL0 and mvL1, the reference indices refIdxL0 and refIdxL1, and the prediction utilization flags predFlagL0 and predFlagL1 are derived by the following ordered steps:

1. The derivation process for an initial merge candidate list as specified in clause I.8.5.3.2.2 is invoked with the luma location (xCb, yCb), the luma location (xPb, yPb), the variables nCbS, nPbW, nPbH, and the partition index partIdx as inputs, and the outputs being a modified luma location (xPb, yPb), the modified variables nPbW and nPbH, the modified variable partIdx, the luma location (xOrigP, yOrigP), the variables nOrigPbW and nOrigPbH, the merging candidate list initMergeCandList, the luma motion vectors mvL0N and mvL1N, the reference indices refIdxL0N and refIdxL1N, and the prediction list utilization flags predFlagL0N and predFlagL1N, with N being replaced by all elements of initMergeCandList.
2. For N being replaced by A₁, B₁, B₀, A₀ and B₂, the following applies:
 - If N is an element in initMergeCandList, availableFlagN is set equal to 1.
 - Otherwise (N is not an element in initMergeCandList), availableFlagN is set equal to 0.
3. Depending on the values of IvDiMcEnabledFlag, DispAvailFlag, and IlluCompFlag[xCb][yCb], the following applies:
 - If IvDiMcEnabledFlag is equal to 0, DispAvailFlag is equal to 0, or IlluCompFlag[xCb][yCb] is equal to 1, the flags availableFlagIV and availableFlagIVShift are set equal to 0.
 - Otherwise (IvDiMcEnabledFlag is equal to 1, DispAvailFlag is equal to 1, and IlluCompFlag[xCb][yCb] is equal to 0), the derivation process for inter-view predicted merging candidates as specified in clause I.8.5.3.2.8 is invoked with the luma location (xPb, yPb), and the variables nPbW and nPbH as inputs, and the availability flags availableFlagIV and availableFlagIVShift, the prediction list utilization flags predFlagLXIV and predFlagLXIVShift, the reference indices refIdxLXIV and refIdxLXIVShift, and the motion vectors mvLXIV and mvLXIVShift (with X in the range of 0 to 1, inclusive) as outputs.
4. Depending on the value of TexMcEnabledFlag, the texture merging candidate is derived as follows:
 - If TexMcEnabledFlag is equal to 0, the variable availableFlagT is set equal to 0.
 - Otherwise (TexMcEnabledFlag is equal to 1), the following applies:
 - The derivation process for sub-block partition motion vectors for an inter-layer predicted merging candidate as specified in clause I.8.5.3.2.9 is invoked with the luma location (xPb, yPb), the variables nPbW and nPbH, and the merging candidate indicator N equal to T as inputs, and the prediction utilization flag predFlagLXT, the reference index refIdxLXT, and the motion vector mvLXT (with X in the range of 0 to 1, inclusive) as outputs.
 - The flag availableFlagT is set equal to (predFlagL0T || predFlagL1T).
5. Depending on the values of IvDiMcEnabledFlag, DispAvailFlag, and DepthFlag, the following applies:
 - If IvDiMcEnabledFlag is equal to 0, DispAvailFlag is equal to 0, or DepthFlag is equal to 1, the flags availableFlagDI and availableFlagDIShift are set equal to 0.
 - Otherwise (IvDiMcEnabledFlag is equal to 1, DispAvailFlag is equal to 1, and DepthFlag is equal to 0), the derivation process for disparity information merging candidates as specified in clause I.8.5.3.2.12 is invoked with the luma location (xPb, yPb), and the variables nPbW and nPbH as inputs, and the availability flags availableFlagDI and availableFlagDIShift, the prediction list utilization flags predFlagLXDI and predFlagLXDIShift, the reference indices refIdxLXDI and refIdxLXDIShift, and the motion vectors mvLXDI and mvLXDIShift (with X in the range of 0 to 1, inclusive) as outputs.
6. Depending on the values of VspMcEnabledFlag, DispAvailFlag, IlluCompFlag[xCb][yCb], iv_res_pred_weight_idx[xCb][yCb], and DbppFlag[xCb][yCb], the following applies:
 - If one or more of the following conditions are true, the flag availableFlagVSP is set equal to 0:
 - VspMcEnabledFlag is equal to 0.

- DispAvailFlag is equal to 0.
- IlluCompFlag[xCb][yCb] is equal to 1.
- iv_res_pred_weight_idx[xCb][yCb] is not equal to 0.
- DbbpFlag[xCb][yCb] is equal to 1.
- Otherwise, the derivation process for a view synthesis prediction merging candidate as specified in clause I.8.5.3.2.13 is invoked with the luma locations (xPb, yPb) and the variables nPbW and nPbH as inputs, and the availability flag availableFlagVSP, the prediction list utilization flag predFlagLXVSP, the reference index refIdxLXVSP, and the motion vector mvLXVSP (with X in the range of 0 to 1, inclusive) as outputs.

7. The merging candidate list extMergeCandList is constructed as follows:

```

i = 0
if( availableFlagT )
    extMergeCandList[ i++ ] = T
if( availableFlagIV && ( !availableFlagT || differentMotion( T, IV ) ) )
    extMergeCandList[ i++ ] = IV
N = DepthFlag ? T : IV
if( availableFlagA1 && ( !availableFlagN || differentMotion( N, A1 ) ) )
    extMergeCandList[ i++ ] = A1
if( availableFlagB1 && ( !availableFlagN || differentMotion( N, B1 ) ) )
    extMergeCandList[ i++ ] = B1
if( availableFlagVSP && ( !availableFlagA1 || !VspMcFlag[ xPb - 1 ][ yPb + nPbH - 1 ] ) &&
    i < MaxNumMergeCand )
    extMergeCandList[ i++ ] = VSP
if( availableFlagB0 )
    extMergeCandList[ i++ ] = B0
if( availableFlagDI && ( !availableFlagA1 || differentMotion( A1, DI ) ) &&
    ( !availableFlagB1 || differentMotion( B1, DI ) ) && ( i < MaxNumMergeCand ) )
    extMergeCandList[ i++ ] = DI
if( availableFlagA0 && i < MaxNumMergeCand )
    extMergeCandList[ i++ ] = A0
if( availableFlagB2 && i < MaxNumMergeCand )
    extMergeCandList[ i++ ] = B2
if( availableFlagIVShift && i < MaxNumMergeCand &&
    ( !availableFlagIV || differentMotion( IV, IVShift ) ) )
    extMergeCandList[ i++ ] = IVShift
if( availableFlagDIShift && !availableFlagIVShift && i < MaxNumMergeCand )
    extMergeCandList[ i++ ] = DIShift
j = 0
while( i < MaxNumMergeCand ) {
    N = initMergeCandList[ j++ ]
    if( N != A1 && N != B1 && N != B0 && N != A0 && N != B2 )
        extMergeCandList[ i++ ] = N
}

```

8. The variable N is derived as follows:

- If (nOrigPbW + nOrigPbH) is equal to 12, the following applies:

$$N = \text{initMergeCandList}[\text{merge_idx}[\text{xOrigP}][\text{yOrigP}]] \quad (\text{I-137})$$

- Otherwise, ((nOrigPbW + nOrigPbH) is not equal to 12), the following applies:

$$N = \text{extMergeCandList}[\text{merge_idx}[\text{xOrigP}][\text{yOrigP}]] \quad (\text{I-138})$$

9. When N is equal to Col, the following applies for X in the range of 0 to 1, inclusive:

- When AltRefFlagLX is equal to 1, refIdxLXCol is set equal to AltRefIdxLX.

10. When availableFlagVSP is equal to 1, N is equal to A₁, and VspMcFlag[xPb - 1][yPb + nPbH - 1] is equal to 1, N is set equal to VSP.

11. The variable subPbMotionFlag is derived as follows:

$$\text{subPbMotionFlag} = (\text{!DepthFlag} \ \&\& \ \text{!DbbpFlag}[\text{xCb}][\text{yCb}] \ \&\& \ \text{N} == \text{IV}) \\ || \ \text{N} == \text{VSP} || \ \text{N} == \text{T} \quad (\text{I-139})$$

12. Depending on the value of subPbMotionFlag, the following applies:

- If subPbMotionFlag is equal to 0, the following applies, for X in the range of 0 to 1, inclusive:

$$\text{mvLX} = \text{mvLXN} \quad (\text{I-140})$$

$$\text{refIdxLX} = \text{refIdxLXN} \quad (\text{I-141})$$

$$\text{predFlagLX} = \text{predFlagLXN} \quad (\text{I-142})$$

- Otherwise (subPbMotionFlag is equal to 1), SubPbArrayPartSize is set equal to SubPbArrayPartSizeN and the following applies for X in the range of 0 to 1, inclusive:

$$\text{mvLX} = (0, 0) \quad (\text{I-143})$$

$$\text{SubPbArrayMvLX} = \text{SubPbArrayMvLXN} \quad (\text{I-144})$$

$$\text{SubPbArrayMvCLX} = \text{SubPbArrayMvCLXN} \quad (\text{I-145})$$

$$\text{refIdxLX} = -1 \quad (\text{I-146})$$

$$\text{SubPbArrayRefIdxLX} = \text{SubPbArrayRefIdxLXN} \quad (\text{I-147})$$

$$\text{predFlagLX} = 0 \quad (\text{I-148})$$

$$\text{SubPbArrayPredFlagLX} = \text{SubPbArrayPredFlagLXN} \quad (\text{I-149})$$

NOTE – When subPbMotionFlag is equal to 1, luma motion vectors, chroma motion vectors, reference indices, and prediction utilization flags are given in sub-block partition granularity in the arrays SubPbArrayPredFlagLX, SubPbArrayMvLX, SubPbArrayMvCLX, SubPbArrayRefIdxLX (with X in the range of 0 to 1, inclusive). Moreover, the width and height of the sub-block partitions is stored in the array SubPbArrayPartSize.

13. When all of the following conditions are true, refIdxL1 is set equal to -1 and predFlagL1 is set equal to 0:

- predFlagL0 and predFlagL1 are equal to 1.
- (nOrigPbW + nOrigPbH) is equal to 12 or DbbpFlag[xCb][yCb] is equal to 1.

14. The flag ivMcFlag is set equal to (N == IV || N == IVShift).

15. The flag vspMcFlag is set equal to (N == VSP).

I.8.5.3.2.8 Derivation process for inter-view predicted merging candidates

Inputs to this process are:

- a luma location (xPb, yPb) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
- two variables nPbW and nPbH specifying the width and the height of the current luma prediction block.

Outputs of this process are (with X in the range of 0 to 1, inclusive):

- the availability flags availableFlagIV and availableFlagIVShift specifying whether the inter-view predicted and shifted inter-view predicted merging candidates are available,
- the prediction list utilization flags predFlagLXIV and predFlagLXIVShift,
- the reference indices refIdxLXIV and refIdxLXIVShift,
- the motion vectors mvLXIV and mvLXIVShift.

The availability flags availableFlagIV and availableFlagIVShift are set equal to 0, and for X in the range of 0 to 1, inclusive, the following applies:

- The variables predFlagLXIV and predFlagLXIVShift are set equal to 0, the variables refIdxLXIV and refIdxLXIVShift are set equal to -1, and the motion vectors mvLXIV and mvLXIVShift are set equal to (0, 0).

The inter-view predicted merging candidate is derived by the following ordered steps:

1. Depending on the values of DbbpFlag[xPb][yPb] and DepthFlag, the following applies:
 - If DbbpFlag[xPb][yPb] is equal to 0 and DepthFlag is equal to 0, the derivation process for sub-block partition motion vectors for an inter-layer predicted merging candidate as specified in clause I.8.5.3.2.9 is

invoked with the luma location (xPb, yPb), the variables nPbW and nPbH, and the merging candidate indicator N equal to IV as inputs, and the outputs are the flag predFlagLXIV, the reference index refIdxLXIV, and the motion vector mvLXIV (with X in the range of 0 to 1, inclusive).

- Otherwise (DbbpFlag[xPb][yPb] is not equal to 0 or DepthFlag is equal to 1), the derivation process for motion vectors for an inter-view predicted merging candidate as specified in clause I.8.5.3.2.10 is invoked with the luma location (xPb, yPb), the variables nPbW and nPbH, and the disparity vector offset (xOff, yOff) equal to (0, 0) as inputs, and the outputs are the flag predFlagLXIV, the reference index refIdxLXIV, and the motion vector mvLXIV (with X in the range of 0 to 1, inclusive).

2. The availability flag availableFlagIV is set equal to (predFlagL0IV || predFlagL1IV).

When DepthFlag is equal to 0, the shifted inter-view predicted merging candidate is derived by the following ordered steps:

1. The derivation process for motion vectors for an inter-view predicted merging candidate as specified in clause I.8.5.3.2.10 is invoked with the luma location (xPb, yPb), the variables nPbW and nPbH, and the disparity vector offset (xOff, yOff) equal to (nPbW * 2, nPbH * 2) as inputs, and the outputs are the flag predFlagLXIVShift, the reference index refIdxLXIVShift, and the motion vector mvLXIVShift (with X in the range of 0 to 1, inclusive).
2. The availability flag availableFlagIVShift is set equal to (predFlagL0IVShift || predFlagL1IVShift).

I.8.5.3.2.9 Derivation process for sub-block partition motion vectors for an inter-layer predicted merging candidate

Inputs to this process are:

- a luma location (xPb, yPb) of the top-left sample of the current prediction luma block relative to the top-left luma sample of the current picture,
- two variables nPbW and nPbH specifying the width and the height of the current prediction block,
- a merging candidate indicator N.

Outputs of this process are (with X in the range of 0 to 1, inclusive):

- the prediction utilization flag predFlagLXN,
- the reference index refIdxLXN,
- the motion vector mvLXN.

For X in the range of 0 to 1, inclusive, the variable predFlagLXN is set equal to 0, the variable refIdxLXN is set equal to -1, the motion vector mvLXN is set equal to (0, 0).

The variables nSbW and nSbH specifying the width and height of the sub-block partitions and the luma location (xSbDef, ySbDef) of the default sub-block partition are derived as follows:

$$\text{subPbTmcSize} = 1 \ll (\log_2_texmc_sub_pb_size_minus3[\text{DepthFlag}] + 3) \quad (\text{I-150})$$

$$\text{subPbIvMcSize} = 1 \ll (\log_2_ivmc_sub_pb_size_minus3[\text{DepthFlag}] + 3) \quad (\text{I-151})$$

$$\text{minSize} = (N == T) ? \text{subPbTmcSize} : \text{subPbIvMcSize} \quad (\text{I-152})$$

$$\text{nSbW} = (\text{nPbW} \% \text{minSize} != 0 \mid \mid \text{nPbH} \% \text{minSize} != 0) ? \text{nPbW} : \text{minSize} \quad (\text{I-153})$$

$$\text{nSbH} = (\text{nPbW} \% \text{minSize} != 0 \mid \mid \text{nPbH} \% \text{minSize} != 0) ? \text{nPbH} : \text{minSize} \quad (\text{I-154})$$

$$(\text{xSbDef}, \text{ySbDef}) = (\text{xPb} + ((\text{nPbW} / \text{nSbW}) / 2) * \text{nSbW}, \text{yPb} + ((\text{nPbH} / \text{nSbH}) / 2) * \text{nSbH}) \quad (\text{I-155})$$

Depending on the value of N, the variables predFlagLXN, refIdxLXN, and mvLXN are derived as follows:

- If N is equal to IV, the derivation process for motion vectors for an inter-view predicted merging candidate as specified in clause I.8.5.3.2.10 is invoked with the luma location (xSbDef, ySbDef), the variables nSbW and nSbH, and the disparity vector offset (xOff, yOff) equal to (0, 0) as inputs, and the outputs are the flag predFlagLXN, the reference index refIdxLXN, and the motion vector mvLXN (with X in the range of 0 to 1, inclusive).
- Otherwise (N is equal to T), the derivation process for motion vectors for the texture merge candidate as specified in clause I.8.5.3.2.11 is invoked with the luma location (xSbDef, ySbDef), and the variables nSbW and nSbH as inputs, and the outputs are the flags predFlagLXN, the reference index refIdxLXN, and the motion vector mvLXN (with X in the range of 0 to 1, inclusive).

When predFlagL0N or predFlagL1N is equal to 1, the following applies:

- For $yBlk$ in the range of 0 to $(nPbH / nSbH - 1)$, inclusive, the following applies:
 - For $xBlk$ in the range of 0 to $(nPbW / nSbW - 1)$, inclusive, the following applies:
 - The luma location (xSb, ySb) of the current sub-block partition is derived as follows:

$$(xSb, ySb) = (xPb + xBlk * nSbW, yPb + yBlk * nSbH) \quad (I-156)$$
 - Depending on the value of N , the following applies:
 - If N is equal to IV, the derivation process for motion vectors for an inter-view predicted merging candidate as specified in clause I.8.5.3.2.10 is invoked with the luma location (xSb, ySb) , the variables $nSbW$ and $nSbH$, and the disparity vector offset $(xOff, yOff)$ equal to $(0, 0)$ as inputs, and the outputs are the flag $spPredFlagLX[xBlk][yBlk]$, the reference index $spRefIdxLX[xBlk][yBlk]$, and the motion vector $spMvLX[xBlk][yBlk]$ (with X in the range of 0 to 1, inclusive).
 - Otherwise (N is equal to T), the derivation process for motion vectors for the texture merge candidate as specified in clause I.8.5.3.2.11 is invoked with the luma location (xSb, ySb) , and the variables $nSbW$ and $nSbH$ as inputs, and the outputs are the flags $spPredFlagLX[xBlk][yBlk]$, the reference index $spRefIdxLX[xBlk][yBlk]$, and the motion vector $spMvLX[xBlk][yBlk]$ (with X in the range of 0 to 1, inclusive).
 - When $spPredFlagL0[xBlk][yBlk]$ and $spPredFlagL1[xBlk][yBlk]$ are both equal to 0, the following applies for X in the range of 0 to 1, inclusive:

$$spPredFlagLX[xBlk][yBlk] = predFlagLXN \quad (I-157)$$

$$spRefIdxLX[xBlk][yBlk] = refIdxLXN \quad (I-158)$$

$$spMvLX[xBlk][yBlk] = mvLXN \quad (I-159)$$
- For use in derivation processes of variables invoked later in the decoding process, the following assignments are made for $x = 0..nPbW - 1$ and $y = 0..nPbH - 1$:
 - The variable $SubPbArrayPartSizeN$ is derived as follows:

$$SubPbArrayPartSizeN[xPb + x][yPb + y] = (nSbW, nSbH) \quad (I-160)$$
 - For X in the range of 0 to 1, inclusive, the following applies:
 - The variables $SubPbArrayPredFlagLX$, $SubPbArrayMvLX$, and $SubPbArrayRefIdxLX$ are derived as follows:

$$SubPbArrayPredFlagLXN[xPb + x][yPb + y] = spPredFlagLX[x / nSbW][y / nSbH] \quad (I-161)$$

$$SubPbArrayRefIdxLXN[xPb + x][yPb + y] = spRefIdxLX[x / nSbW][y / nSbH] \quad (I-162)$$

$$SubPbArrayMvLXN[xPb + x][yPb + y] = spMvLX[x / nSbW][y / nSbH] \quad (I-163)$$
 - The derivation process for chroma motion vectors as specified in clause 8.5.3.2.10 is invoked with $SubPbArrayMvLXN[xPb + x][yPb + y]$ as input, and the output is $SubPbArrayMvCLXN[xPb + x][yPb + y]$.

I.8.5.3.2.10 Derivation process for motion vectors for an inter-view predicted merging candidate

Inputs to this process are:

- a luma location $(xBlk, yBlk)$ of the top-left sample of the current luma prediction block or sub-block partition relative to the top-left luma sample of the current picture,
- two variables $nBlkW$ and $nBlkH$ specifying the width and the height of the current luma prediction block or sub-block partition,
- a disparity vector offset $(xOff, yOff)$.

Outputs of this process are (with X in the range of 0 to 1, inclusive):

- the prediction utilization flag $predFlagLXInterView$,
- the reference index $refIdxLXInterView$,
- the motion vector $mvLXInterView$.

For X in the range of 0 to 1, inclusive, the flag $predFlagLXInterView$ is set equal to 0, the variable $refIdxLXInterView$ is set equal to -1 , and the motion vector $mvLXInterView$ is set equal to $(0, 0)$.

The luma location (xRef, yRef) is derived as follows:

$$\text{dispVec}[0] = \text{DispRefVec}[x_{\text{Blk}}][y_{\text{Blk}}][0] + x_{\text{Off}} \quad (\text{I-164})$$

$$\text{dispVec}[1] = \text{DispRefVec}[x_{\text{Blk}}][y_{\text{Blk}}][1] + y_{\text{Off}} \quad (\text{I-165})$$

$$x_{\text{RefFull}} = x_{\text{Blk}} + (n_{\text{BlkW}} \gg 1) + ((\text{dispVec}[0] + 2) \gg 2) \quad (\text{I-166})$$

$$y_{\text{RefFull}} = y_{\text{Blk}} + (n_{\text{BlkH}} \gg 1) + ((\text{dispVec}[1] + 2) \gg 2) \quad (\text{I-167})$$

$$x_{\text{Ref}} = \text{Clip3}(0, \text{pic_width_in_luma_samples} - 1, (x_{\text{RefFull}} \gg 3) \ll 3) \quad (\text{I-168})$$

$$y_{\text{Ref}} = \text{Clip3}(0, \text{pic_height_in_luma_samples} - 1, (y_{\text{RefFull}} \gg 3) \ll 3) \quad (\text{I-169})$$

Let ivRefPic the picture with nuh_layer_id equal to ViewCompLayerId[RefViewIdx[xBlk][yBlk]][DepthFlag] in the current access unit and let ivRefPb be the luma prediction block covering the luma location (xRef, yRef) in the picture ivRefPic.

The luma location (xIvRefPb, yIvRefPb) is set equal to the location of the top-left sample of ivRefPb relative to the top-left luma sample of the picture ivRefPic.

When ivRefPb is not coded in an intra prediction mode, the following applies for X in the range of 0 to (slice_type == B) ? 1 : 0, inclusive:

- For k in the range of 0 to 1, inclusive, the following applies:
 - Y is set equal to (k == 0) ? X : (1 – X).
 - The variables predFlagLYIvRef[x][y], mvLYIvRef[x][y], and refIdxLYIvRef[x][y] are set equal to PredFlagLY[x][y], MvLY[x][y], and RefIdxLY[x][y], respectively, of picture ivRefPic.
 - The variable refPicListYIvRef is set equal to RefPicListY of the slice containing ivRefPb in picture ivRefPic.
 - When predFlagLYIvRef[xIvRefPb][yIvRefPb] is equal to 1, the following applies for i in the range of 0 to num_ref_idx_LX_active_minus1, inclusive:
 - When PicOrderCnt(refPicListYIvRef[refIdxLYIvRef[xIvRefPb][yIvRefPb]]) is equal to PicOrderCnt(RefPicListX[i]) and predFlagLXInterView is equal to 0, the following applies:

$$\text{predFlagLXInterView} = 1 \quad (\text{I-170})$$

$$\text{refIdxLXInterView} = i \quad (\text{I-171})$$

$$\text{mvLXInterView} = \text{mvLYIvRef}[x_{\text{IvRefPb}}][y_{\text{IvRefPb}}] \quad (\text{I-172})$$

I.8.5.3.2.11 Derivation process for motion vectors for the texture merge candidate

Inputs to this process are:

- a luma location (xSb, ySb) of the top-left sample of the current luma sub-block partition relative to the top-left luma sample of the current picture,
- two variables nSbW and nSbH specifying the width and the height of the current sub-block partition,

Outputs of this process are (with X in the range of 0 to 1, inclusive):

- the prediction utilization flag predFlagLXT,
- the reference index refIdxLXT,
- the motion vector mvLXT.

For X in the range of 0 to 1, inclusive, the variable predFlagLXT is set equal to 0, the variable refIdxLXT is set equal to –1, and the motion vector mvLXT is set equal to (0, 0).

The texture luma location (xRef, yRef) is derived as follows:

$$x_{\text{RefFull}} = x_{\text{Sb}} + (n_{\text{SbW}} \gg 1) \quad (\text{I-173})$$

$$y_{\text{RefFull}} = y_{\text{Sb}} + (n_{\text{SbH}} \gg 1) \quad (\text{I-174})$$

$$x_{\text{Ref}} = (x_{\text{RefFull}} \gg 3) \ll 3 \quad (\text{I-175})$$

$$y_{\text{Ref}} = (y_{\text{RefFull}} \gg 3) \ll 3 \quad (\text{I-176})$$

Let textPb be the prediction block covering the luma sample location (xRef, yRef) in the picture TexturePic.

For X in the range of 0 to $(\text{slice_type} == B) ? 1 : 0$, inclusive, the following applies:

- The arrays $\text{textPredFlagLX}[x][y]$, $\text{textRefIdxLX}[x][y]$, and $\text{textMvLX}[x][y]$ are set equal to $\text{PredFlagLX}[x][y]$, $\text{RefIdxLX}[x][y]$, and $\text{MvLX}[x][y]$, respectively, of the picture TexturePic .
- The list textRefPicListX is set equal to RefPicListX of the slice containing textPb in the picture TexturePic .
- When $\text{textPredFlagLX}[\text{xRef}][\text{yRef}]$ is equal to 1, the following applies for i in the range of 0 to $\text{num_ref_idx_lX_active_minus1}$, inclusive:
 - When $\text{PicOrderCnt}(\text{RefPicListX}[i])$ is equal to $\text{PicOrderCnt}(\text{textRefPicListX}[\text{textRefIdxLX}])$, $\text{ViewIdx}(\text{RefPicListX}[i])$ is equal to $\text{ViewIdx}(\text{textRefPicListX}[\text{textRefIdxLX}])$, and predFlagLXT is equal to 0, the following applies:

$$\text{predFlagLXT} = 1 \quad (\text{I-177})$$

$$\text{refIdxLXT} = i \quad (\text{I-178})$$

$$\text{mvLXT}[0] = (\text{textMvLX}[\text{xRef}][\text{yRef}][0] + 2) \gg 2 \quad (\text{I-179})$$

$$\text{mvLXT}[1] = (\text{textMvLX}[\text{xRef}][\text{yRef}][1] + 2) \gg 2 \quad (\text{I-180})$$

1.8.5.3.2.12 Derivation process for disparity information merging candidates

Inputs to this process are:

- a luma location (xPb, yPb) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
- two variables nPbW and nPbH specifying the width and the height of the current prediction block.

Outputs of this process are (with X in the range of 0 to 1, inclusive):

- the flags availableFlagDI and $\text{availableFlagDIShift}$, specifying whether the disparity information merging candidate and the shifted disparity information candidate are available,
- the prediction list utilization flags predFlagLXDI and predFlagLXDISHift ,
- the reference indices refIdxLXDI and refIdxLXDISHift ,
- the motion vectors mvLXDI and mvLXDISHift .

The variables availableFlagDI and $\text{availableFlagDIShift}$ are set equal to 0, and for X in the range of 0 to 1, inclusive, the following applies:

- The variables predFlagLXDI and predFlagLXDISHift are set equal to 0, the variables refIdxLXDI and refIdxLXDISHift are set equal to -1, and the motion vectors mvLXDI and mvLXDISHift are set equal to $(0, 0)$.

The disparity information merging candidate is derived as follows:

- For X in the range of 0 to $(\text{slice_type} == B) ? 1 : 0$, inclusive, the following applies:
 - For i in the range of 0 to $\text{num_ref_idx_lX_active_minus1}$, inclusive, the following applies:
 - When $\text{PicOrderCnt}(\text{RefPicListX}[i])$ is equal to the PicOrderCntVal , $\text{ViewIdx}(\text{RefPicListX}[i])$ is equal to $\text{RefViewIdx}[\text{xPb}][\text{yPb}]$, and predFlagLXDI is equal to 0, the following applies:

$$\text{availableFlagDI} = 1 \quad (\text{I-181})$$

$$\text{predFlagLXDI} = 1 \quad (\text{I-182})$$

$$\text{refIdxLXDI} = i \quad (\text{I-183})$$

$$\text{mvLXDI} = (\text{DispRefVec}[\text{xPb}][\text{yPb}][0], 0) \quad (\text{I-184})$$

When availableFlagDI is equal to 1, the shifted disparity information merging candidate is derived as follows:

- The variable $\text{availableFlagDIShift}$ is set equal to 1.
- The following applies for X in the range of 0 to 1, inclusive:

$$\text{predFlagLXDISHift} = \text{predFlagLXDI} \quad (\text{I-185})$$

$$\text{refIdxLXDISHift} = \text{refIdxLXDI} \quad (\text{I-186})$$

$$\text{mvLXDISHift} = \text{predFlagLXDI} ? (\text{mvLXDI}[0] + 4, \text{mvLXDI}[1]) : (0, 0) \quad (\text{I-187})$$

I.8.5.3.2.13 Derivation process for a view synthesis prediction merging candidate

Inputs to this process are:

- a luma location (xPb, yPb) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
- two variables $nPbW$ and $nPbH$ specifying the width and the height of the current prediction block.

Outputs of this process are (with X in the range of 0 to 1, inclusive):

- the availability flag $availableFlagVSP$ specifying whether the view synthesis prediction merging candidate is available,
- the prediction list utilization flag $predFlagLXVSP$,
- the reference index $refIdxLXVSP$,
- the motion vector $mvLXVSP$.

The variable $availableFlagVSP$ is set equal to 0 and for X in the range of 0 to 1, inclusive, the following applies:

- The variable $predFlagLXVSP$ is set equal to 0, the variable $refIdxLXVSP$ is set equal to -1 , and the motion vector $mvLXVSP$ is set equal to (0, 0).

For X in the range of 0 to ($slice_type == B ? 1 : 0$), inclusive, the following applies:

- For i in the range of 0 to $num_ref_idx_IX_active_minus1$, inclusive, the following applies:
 - When $availableFlagVSP$ is equal to 0 and $ViewIdx(RefPicListX[i])$ is equal to $RefViewIdx[xPb][yPb]$, the following applies:

$$availableFlagVSP = 1 \quad (I-188)$$

$$predFlagLXVSP = 1 \quad (I-189)$$

$$refIdxLXVSP = i \quad (I-190)$$

$$mvLXVSP = DispVec[xPb][yPb] \quad (I-191)$$

When $availableFlagVSP$ is equal to 1 the following applies:

- The derivation process for a depth or disparity sample array from a depth picture as specified in clause I.8.5.7 is invoked with the luma location (xPb, yPb), the variables $nPbW$ and $nPbH$, the disparity vector $DispVec[xPb][yPb]$, the reference view order index $RefViewIdx[xPb][yPb]$, and the variable $partIdx$ equal to 2 as inputs, and the outputs are the array $disparitySamples$ of size $(nPbW) \times (nPbH)$, and the flag $horSplitFlag$.
- For use in derivation processes of variables invoked later in the decoding process, the following assignments are made for $x = 0..nPbW - 1$ and $y = 0..nPbH - 1$:

- The variable $SubPbArrayPartSizeVSP[xPb + x][yPb + y]$ is set equal to ($horSplitFlag ? (8, 4) : (4, 8)$).

- For X in the range of 0 to 1, inclusive, the following applies:

- The variables $SubPbArrayPredFlagLXVSP$, $SubPbArrayMvLXVSP$, and $SubPbArrayRefIdxLXVSP$ are derived as follows:

$$SubPbArrayPredFlagLXVSP[xPb + x][yPb + y] = predFlagLXVSP \quad (I-192)$$

$$SubPbArrayRefIdxLXVSP[xPb + x][yPb + y] = refIdxLXVSP \quad (I-193)$$

$$SubPbArrayMvLXVSP[xPb + x][yPb + y] = predFlagLXVSP ? (disparitySamples[x][y], 0) : (0, 0) \quad (I-194)$$

- When $ChromaArrayType$ is not equal to 0, the derivation process for chroma motion vectors as specified in clause 8.5.3.2.9 is invoked with $SubPbArrayMvLXVSP[xPb + x][yPb + y]$ as input, and the output is $SubPbArrayMvCLXVSP[xPb + x][yPb + y]$.

I.8.5.3.3 Decoding process for inter prediction samples

I.8.5.3.3.1 General

Inputs to this process are:

- a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

- a luma location (xBl , yBl) specifying the top-left sample of the current luma prediction block relative to the top-left sample of the current luma coding block,
- a variable $nCbS$ specifying the size of the current luma coding block,
- two variables $nPbW$ and $nPbH$ specifying the width and the height of the luma prediction block,
- the luma motion vectors $mvL0$ and $mvL1$,
- when $ChromaArrayType$ is not equal to 0, the chroma motion vectors $mvCL0$ and $mvCL1$,
- the reference indices $refIdxL0$ and $refIdxL1$,
- the prediction list utilization flags, $predFlagL0$ and $predFlagL1$.

Outputs of this process are:

- an $(nCbS_L) \times (nCbS_L)$ array $predSamples_L$ of luma prediction samples, where $nCbS_L$ is derived as specified below,
- when $ChromaArrayType$ is not equal to 0, an $(nCbSw_C) \times (nCbSh_C)$ array $predSamples_{Cb}$ of chroma prediction samples for the component Cb , where $nCbSw_C$ and $nCbSh_C$ are derived as specified below,
- when $ChromaArrayType$ is not equal to 0, an $(nCbSw_C) \times (nCbSh_C)$ array $predSamples_{Cr}$ of chroma prediction samples for the component Cr , where $nCbSw_C$ and $nCbSh_C$ are derived as specified below.

When $DepthFlag$ is equal to 1, the following applies, for X in the range of 0 to 1, inclusive:

- The variable $mvLX$ is set equal to ($mvLX \ll 2$).
- When $ChromaArrayType$ is not equal to 0, the variable $mvCLX$ is set equal to ($mvCLX \ll 2$).

The variable $nCbS_L$ is set equal to $nCbS$. When $ChromaArrayType$ is not equal to 0, the variable $nCbSw_C$ is set equal to $nCbS / SubWidthC$ and the variable $nCbSh_C$ is set equal to $nCbS / SubHeightC$.

1. Let $predSamplesL0_L$ and $predSamplesL1_L$ be $(nPbW) \times (nPbH)$ arrays of predicted luma sample values and, when $ChromaArrayType$ is not equal to 0, $predSamplesL0_{Cb}$, $predSamplesL1_{Cb}$, $predSamplesL0_{Cr}$, and $predSamplesL1_{Cr}$ be $(nPbW / SubWidthC) \times (nPbH / SubHeightC)$ arrays of predicted chroma sample values.
2. For X in the range of 0 to 1, inclusive, when $predFlagLX$ is equal to 1, the following applies:
 - When $predFlagLX$ is equal to 1, the following applies:
 - If $iv_res_pred_weight_idx[xCb][yCb]$ is not equal to 0, the bilinear sample interpolation and residual prediction process as specified in clause I.8.5.3.3.3 is invoked with the luma locations (xCb , yCb), (xBl , yBl), the size of the current luma coding block $nCbS$, the width and the height of the current luma prediction block $nPbW$ and $nPbH$, the prediction list utilization flags $predFlagL0$ and $predFlagL1$, the reference indices $refIdxL0$ and $refIdxL1$, the motion vectors $mvL0$ and $mvL1$, when $ChromaArrayType$ is not equal to 0, the motion vectors $mvCL0$ and $mvCL1$, and the prediction list indication X as inputs, and the array $predSamplesLX_L$ and, when $ChromaArrayType$ is not equal to 0, the arrays $predSamplesLX_{Cb}$ and $predSamplesLX_{Cr}$, as outputs.
 - Otherwise ($iv_res_pred_weight_idx[xCb][yCb]$ is equal to 0), the following applies:
 - The reference picture consisting of an ordered two-dimensional array $refPicLX_L$ of luma samples and, when $ChromaArrayType$ is not equal to 0, two ordered two-dimensional arrays $refPicLX_{Cb}$ and $refPicLX_{Cr}$ of chroma samples is derived by invoking the process specified in clause 8.5.3.3.2 with $refIdxLX$ as input.
 - The array $predSamplesLX_L$ and, when $ChromaArrayType$ is not equal to 0, the arrays $predSamplesLX_{Cb}$ and $predSamplesLX_{Cr}$ are derived by invoking the fractional sample interpolation process specified in clause 8.5.3.3.3 with the luma locations (xCb , yCb) and (xBl , yBl), the luma prediction block width $nPbW$, the luma prediction block height $nPbH$, the motion vectors $mvLX$ and, when $ChromaArrayType$ is not equal to 0, $mvCLX$, and the reference arrays $refPicLX_L$, $refPicLX_{Cb}$, and $refPicLX_{Cr}$ as inputs.
3. Depending on the value of $IlluCompFlag[xCb][yCb]$, the array $predSamples_L$ is derived as follows:
 - If $IlluCompFlag[xCb][yCb]$ is equal to 0, the following applies:
 - The prediction samples inside the current luma prediction block, $predSamples_L[x_L + xBl][y_L + yBl]$ with $x_L = 0..nPbW - 1$ and $y_L = 0..nPbH - 1$, are derived by invoking the weighted sample prediction process specified in clause 8.5.3.3.4 with the prediction block width $nPbW$, the prediction block height $nPbH$, and the sample arrays $predSamplesL0_L$ and $predSamplesL1_L$, and the variables $predFlagL0$, $predFlagL1$, $refIdxL0$, $refIdxL1$, and $cIdx$ equal to 0 as inputs.

- Otherwise ($\text{IlluCompFlag}[x_{Cb}][y_{Cb}]$ is equal to 1), the following applies:
 - The prediction samples inside the current luma prediction block, $\text{predSamples}_L[x_L + x_{Bl}][y_L + y_{Bl}]$ with $x_L = 0..n_{PbW} - 1$ and $y_L = 0..n_{PbH} - 1$, are derived by invoking the illumination compensated sample prediction process specified in clause I.8.5.3.3.2, with the luma location (x_{Cb}, y_{Cb}), the size of the current luma coding block n_{CbS} , the luma location (x_{Bl}, y_{Bl}), the width and the height of the current luma prediction block n_{PbW} and n_{PbH} , the sample arrays $\text{predSamples}_{L0_L}$ and $\text{predSamples}_{L1_L}$, the variables predFlag_{L0} , predFlag_{L1} , refIdx_{L0} , refIdx_{L1} , mv_{L0} , mv_{L1} and $cIdx$ equal to 0 as inputs.
- 4. When ChromaArrayType is not equal to 0, depending on the values of $\text{IlluCompFlag}[x_{Cb}][y_{Cb}]$ and n_{PbW} , the arrays predSamples_{Cb} , and predSamples_{Cr} are derived as follows:
 - If $\text{IlluCompFlag}[x_{Cb}][y_{Cb}]$ is equal to 0 or n_{PbW} is less than or equal to 8, the following applies:
 - The prediction samples inside the current chroma component Cb prediction block, $\text{predSamples}_{Cb}[x_C + x_{Bl} / \text{SubWidthC}][y_C + y_{Bl} / \text{SubHeightC}]$ with $x_C = 0..n_{PbW} / \text{SubWidthC} - 1$ and $y_C = 0..n_{PbH} / \text{SubHeightC} - 1$, are derived by invoking the weighted sample prediction process specified in clause 8.5.3.3.4 with the prediction block width n_{PbW} set equal to $n_{PbW} / \text{SubWidthC}$, the prediction block height n_{PbH} set equal to $n_{PbH} / \text{SubHeightC}$, the sample arrays $\text{predSamples}_{L0_{Cb}}$ and $\text{predSamples}_{L1_{Cb}}$, and the variables predFlag_{L0} , predFlag_{L1} , refIdx_{L0} , refIdx_{L1} , and $cIdx$ equal to 1 as inputs.
 - The prediction samples inside the current chroma component Cr prediction block, $\text{predSamples}_{Cr}[x_C + x_{Bl} / \text{SubWidthC}][y_C + y_{Bl} / \text{SubHeightC}]$ with $x_C = 0..n_{PbW} / \text{SubWidthC} - 1$ and $y_C = 0..n_{PbH} / \text{SubHeightC} - 1$, are derived by invoking the weighted sample prediction process specified in clause 8.5.3.3.4 with the prediction block width n_{PbW} set equal to $n_{PbW} / \text{SubWidthC}$, the prediction block height n_{PbH} set equal to $n_{PbH} / \text{SubHeightC}$, the sample arrays $\text{predSamples}_{L0_{Cr}}$ and $\text{predSamples}_{L1_{Cr}}$, and the variables predFlag_{L0} , predFlag_{L1} , refIdx_{L0} , refIdx_{L1} , and $cIdx$ equal to 2 as inputs.
 - Otherwise ($\text{IlluCompFlag}[x_{Cb}][y_{Cb}]$ is equal to 1 and n_{PbW} is greater than 8), the following applies:
 - The prediction samples inside the current chroma component Cb prediction block, $\text{predSamples}_{Cb}[x_C + x_{Bl} / \text{SubWidthC}][y_C + y_{Bl} / \text{SubHeightC}]$ with $x_C = 0..n_{PbW} / \text{SubWidthC} - 1$ and $y_C = 0..n_{PbH} / \text{SubHeightC} - 1$, are derived by invoking the illumination compensated sample prediction process specified in clause I.8.5.3.3.2, with the luma location (x_{Cb}, y_{Cb}), the size of the current luma coding block n_{CbS} , the chroma location ($x_{Bl} / \text{SubWidthC}, y_{Bl} / \text{SubHeightC}$), the width and the height of the current chroma prediction block ($n_{PbW} / \text{SubWidthC}$) and ($n_{PbH} / \text{SubHeightC}$), the sample arrays $\text{predSamples}_{L0_{Cb}}$ and $\text{predSamples}_{L1_{Cb}}$, the variables predFlag_{L0} , predFlag_{L1} , refIdx_{L0} , refIdx_{L1} , mv_{CL0} , mv_{CL1} , and $cIdx$ equal to 1 as inputs.
 - The prediction samples inside the current chroma component Cr prediction block, $\text{predSamples}_{Cr}[x_C + x_{Bl} / \text{SubWidthC}][y_C + y_{Bl} / \text{SubHeightC}]$ with $x_C = 0..n_{PbW} / \text{SubWidthC} - 1$ and $y_C = 0..n_{PbH} / \text{SubHeightC} - 1$, are derived by invoking the illumination compensated sample prediction process specified in clause I.8.5.3.3.2, with the luma location (x_{Cb}, y_{Cb}), the size of the current luma coding block n_{CbS} , the chroma location ($x_{Bl} / \text{SubWidthC}, y_{Bl} / \text{SubHeightC}$), the width and the height of the current chroma prediction block ($n_{PbW} / \text{SubWidthC}$) and ($n_{PbH} / \text{SubHeightC}$), the sample arrays $\text{predSamples}_{L0_{Cr}}$ and $\text{predSamples}_{L1_{Cr}}$, the variables predFlag_{L0} , predFlag_{L1} , refIdx_{L0} , refIdx_{L1} , mv_{CL0} , mv_{CL1} , and $cIdx$ equal to 2 as inputs.

I.8.5.3.3.2 Illumination compensated sample prediction process

I.8.5.3.3.2.1 General

Inputs to this process are:

- a location (x_{Cb}, y_{Cb}) specifying the top-left sample of the current luma coding block relative to the top-left sample of the current picture,
- the size of current luma coding block n_{CbS} ,
- a location (x_{Bl}, y_{Bl}) specifying the top-left sample of the current luma or chroma prediction block relative to the top-left sample of the current luma coding block,
- two variables n_{PbW} and n_{PbH} specifying the width and height of the current luma or chroma prediction block,
- two $(n_{PbW} \times n_{PbH})$ arrays predSamples_{L0} and predSamples_{L1} ,
- two prediction list utilization flags predFlag_{L0} and predFlag_{L1} ,

- two reference indices refIdxL0 and refIdxL1,
- two motion vector mvL0 and mvL1,
- a colour component index cIdx.

Output of this process is an (nPbW)x(nPbH) array predSamples of prediction sample values.

The variables bitDepth, shift1, shift2, offset1, and offset2 are derived as follows:

$$\text{bitDepth} = (\text{cIdx} == 0) ? \text{BitDepth}_Y : \text{BitDepth}_C \quad (\text{I-195})$$

$$\text{shift1} = \text{Max}(2, 14 - \text{bitDepth}) \quad (\text{I-196})$$

$$\text{shift2} = \text{Max}(3, 15 - \text{bitDepth}) \quad (\text{I-197})$$

$$\text{offset1} = 1 \ll (\text{shift1} - 1) \quad (\text{I-198})$$

$$\text{offset2} = 1 \ll (\text{shift2} - 1) \quad (\text{I-199})$$

The derivation process for illumination compensation mode availability and parameters as specified in clause I.8.5.3.3.2.2 is invoked with the luma location (xCb, yCb), the size of the current luma coding block nCbS, the prediction list utilization flags predFlagL0 and predFlagL1, the reference indices refIdxL0 and refIdxL1, the motion vectors mvL0 and mvL1, the variable bitDepth, and the variable cIdx as inputs, and the outputs are the flags puIcFlagL0 and puIcFlagL1, the variables icWeightL0 and icWeightL1, and the variables icOffsetL0 and icOffsetL1.

Depending on the value of predFlagL0 and predFlagL1, the prediction samples predSamples[x][y] with $x = 0..(\text{nPbW} - 1)$ and $y = 0..(\text{nPbH} - 1)$ are derived as follows:

- For X in the range of 0 to 1, inclusive, the following applies:

- When predFlagLX is equal to 1, the following applies:

$$\text{clipPredVal} = \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, (\text{predSamplesLX}[x][y] + \text{offset1}) \gg \text{shift1}) \quad (\text{I-200})$$

$$\begin{aligned} \text{predValX} &= !\text{puIcFlagLX} ? \text{clipPredVal} : \\ &(\text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, (\text{clipPredVal} * \text{icWeightLX}) \gg 5) + \text{icOffsetLX}) \end{aligned} \quad (\text{I-201})$$

- If predFlagL0 and predFlagL1 are equal to 1, the following applies:

$$\text{predSamples}[x][y] = \text{Clip3}(0, (1 \ll \text{bitDepth}) - 1, (\text{predVal0} + \text{predVal1} + \text{offset2}) \gg \text{shift2}) \quad (\text{I-202})$$

- Otherwise (predFlagL0 is equal to 0 or predFlagL1 is equal to 0), the following applies:

$$\text{predSamples}[x][y] = \text{predFlagL0} ? \text{predVal0} : \text{predVal1} \quad (\text{I-203})$$

I.8.5.3.3.2.2 Derivation process for illumination compensation mode availability and parameters

Inputs to this process are:

- a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- the size of the current luma coding block nCbS,
- two prediction list utilization flags predFlagL0 and predFlagL1,
- two reference indices refIdxL0 and refIdxL1,
- two motion vectors mvL0 and mvL1,
- a bit depth of samples bitDepth,
- a variable cIdx specifying colour component index.

Outputs of this process are:

- the flags puIcFlagL0 and puIcFlagL1 specifying whether illumination compensation is enabled,
- the variables icWeightL0 and icWeightL1 specifying weights for illumination compensation,
- the variables icOffsetL0 and icOffsetL1 specifying offsets for illumination compensation.

The variables puIcFlagL0 and puIcFlagL1 are set equal to 0, the variables icWeightL0 and icWeightL1 are set equal to 1, and the variables icOffsetL0 and icOffsetL1 are set equal to 0.

The variables subWidth, subHeight, and the location (xC, yC) specifying the top-left sample of the current luma or chroma

coding block are derived as follows:

$$\text{subWidth} = (\text{cIdx} == 0) ? 1 : \text{SubWidthC} \quad (\text{I-204})$$

$$\text{subHeight} = (\text{cIdx} == 0) ? 1 : \text{SubHeightC} \quad (\text{I-205})$$

$$(\text{xC}, \text{yC}) = (\text{xCb} / \text{subWidth}, \text{yCb} / \text{subHeight}) \quad (\text{I-206})$$

The variable `availFlagCurAboveRow` specifying the availability of above neighbouring row samples is derived by invoking the availability derivation process for a block in z-scan order as specified in clause 6.4.1 with the location $(\text{xCurr}, \text{yCurr})$ set equal to (xCb, yCb) and the neighbouring location (xN, yN) set equal to $(\text{xCb}, \text{yCb} - 1)$ as inputs, and the output is assigned to `availFlagCurAboveRow`.

The variable `availFlagCurLeftCol` specifying the availability of left neighbouring column samples is derived by invoking the availability derivation process for a block in z-scan order as specified in clause 6.4.1 with the location $(\text{xCurr}, \text{yCurr})$ set equal to (xCb, yCb) and the neighbouring location (xN, yN) set equal to $(\text{xCb} - 1, \text{yCb})$ as inputs, and the output is assigned to `availFlagCurLeftCol`.

When `availFlagCurAboveRow` is equal to 1 or `availFlagCurLeftCol` is equal to 1, the following applies:

1. Depending on the value of `cIdx`, the variable `curRecSamples` specifying the reconstructed picture samples before deblocking filter of the current picture is derived as follows:

$$\text{curRecSamples} = (\text{cIdx} == 0) ? S_L : ((\text{cIdx} == 1) ? S_{Cb} : S_{Cr}) \quad (\text{I-207})$$

2. For X in the range of 0 to 1, inclusive, when `predFlagLX` is equal to 1, the following applies:

- Let `refPicLX` be the picture `RefPicListX[refIdxLX]`.
- When `ViewIdx(refPicLX)` is not equal to `ViewIdx`, the following applies:
 - The variable `puIcFlagLX` is set equal to 1.
 - The luma location $(\text{xRefBlkLX}, \text{yRefBlkLX})$ specifying the top-left sample of the reference block in the picture `refPicLX` is derived as follows:

$$\text{xRefBlkLX} = \text{xC} + ((\text{mvLX}[0] + (\text{cIdx} ? 4 : 2)) \gg (2 + (\text{cIdx} ? 1 : 0))) \quad (\text{I-208})$$

$$\text{yRefBlkLX} = \text{yC} + ((\text{mvLX}[1] + (\text{cIdx} ? 4 : 2)) \gg (2 + (\text{cIdx} ? 1 : 0))) \quad (\text{I-209})$$

- Depending on the value of `cIdx`, the variable `refRecSamplesLX` specifying the reconstructed picture samples of the picture `refPicLX` is derived as follows:
 - If `cIdx` is equal to 0, `refRecSamplesLX` is set equal to reconstructed picture sample array S_L of picture `refPicLX`.
 - Otherwise, if `cIdx` is equal to 1, `refRecSamplesLX` is set equal to the reconstructed chroma sample array S_{Cb} of picture `refPicLX`.
 - Otherwise (`cIdx` is equal to 2), `refRecSamplesLX` is set equal to the reconstructed chroma sample array S_{Cr} of picture `refPicLX`.
3. The lists `curSampleList`, `refSampleList0`, and `refSampleList1`, specifying the neighbouring samples in pictures `CurrPic`, `refPicL0`, and `refPicL1`, respectively, are derived as follows:
 - The variable `numSamples` specifying the number of elements of `curSampleList`, `refSampleList0`, and `refSampleList1` is set equal to 0.
 - For `curNbColFlag` in the range of 0 to 1, inclusive, the following applies:
 - When $(\text{curNbColFlag} ? \text{availFlagCurLeftCol} : \text{availFlagCurAboveRow})$ is equal to 1, the following applies, for i in the range of 0 to $(\text{nCbS} / (\text{curNbColFlag} ? \text{subHeight} : \text{subWidth})) - 1$, inclusive:
 - The variables `xOff` and `yOff` are derived as follows:

$$\text{xOff} = \text{curNbColFlag} ? -1 : i \quad (\text{I-210})$$

$$\text{yOff} = \text{curNbColFlag} ? i : -1 \quad (\text{I-211})$$

- For X in the range of 0 to 1, inclusive, when `puIcFlagLX` is equal to 1, the following applies:

$$\text{xP} = \text{Clip3}(0, (\text{pic_width_in_luma_samples} / \text{subWidth}) - 1, \text{xRefBlkLX} + \text{xOff}) \quad (\text{I-212})$$

$$\text{yP} = \text{Clip3}(0, (\text{pic_height_in_luma_samples} / \text{subHeight}) - 1, \text{yRefBlkLX} + \text{yOff}) \quad (\text{I-213})$$

$$\text{refSampleListLX}[\text{numSamples}] = \text{refRecSamplesLX}[\text{xP}][\text{yP}] \quad (\text{I-214})$$

- The variables `curSampleList` and `numSamples` are modified as follows:

$$\text{curSampleList}[\text{numSamples}++] = \text{curRecSamples}[\text{xC} + \text{xOff}][\text{yC} + \text{yOff}] \quad (\text{I-215})$$

4. For X in the range of 0 to 1, inclusive, when `puIcFlagLX` is equal to 1, `icWeightLX` and `icOffsetLX` are modified as follows:
 - The derivation process for illumination compensation parameters as specified in clause I.8.5.3.3.2.3 is invoked, with the list of neighbouring samples in the current picture `curSampleList`, the list of neighbouring samples in the reference picture `refSampleListX`, the number of neighbouring samples `numSamples`, the variable `bitDepth`, and the variable `cIdx` as inputs, and the variables `icWeightLX` and `icOffsetLX` as outputs.

I.8.5.3.3.2.3 Derivation process for illumination compensation parameters

Inputs to this process are:

- a list `curSampleList` specifying the current neighbouring samples,
- a list `refSampleList` specifying the reference neighbouring samples,
- a variable `numSamples` specifying the number of elements of `curSampleList` and `refSampleList`,
- a variable `bitDepth` specifying the bit depth of samples,
- a variable `cIdx` specifying colour component index.

Outputs of this process are:

- the variable `icWeight` specifying a weight for illumination compensation,
- the variable `icOffset` specifying an offset for illumination compensation.

The variable `precShift` is set equal to $\text{Max}(0, \text{bitDepth} - 12)$.

The variables `sumRef` and `sumCur` are set equal to 0 and the following applies for i in the range of 0 to $(\text{numSamples} / 2 - 1)$, inclusive:

$$\text{sumRef} += \text{refSampleList}[2 * i] \quad (\text{I-216})$$

$$\text{sumCur} += \text{curSampleList}[2 * i] \quad (\text{I-217})$$

The variables `avgShift` and `avgOffset` are derived as follows:

$$\text{avgShift} = \text{Log2}(\text{numSamples} / 2) \quad (\text{I-218})$$

$$\text{avgOffset} = 1 \ll (\text{avgShift} - 1) \quad (\text{I-219})$$

Depending on the value of `cIdx`, the variables `icWeight` and `icOffset` are derived as follows:

- If `cIdx` is equal to 0, the following applies:
 - The variables `sumRefSquare` and `sumProdRefCur` are set equal to 0, and the following applies for i in the range of 0 to $(\text{numSamples} / 2 - 1)$, inclusive:

$$\text{sumRefSquare} += (\text{refSampleList}[2 * i] * \text{refSampleList}[2 * i]) \gg \text{precShift} \quad (\text{I-220})$$

$$\text{sumProdRefCur} += (\text{refSampleList}[2 * i] * \text{curSampleList}[2 * i]) \gg \text{precShift} \quad (\text{I-221})$$
 - The variables `numerDiv` and `denomDiv` are derived as follows:

$$\text{denomDiv} = ((\text{sumRefSquare} + (\text{sumRefSquare} \gg 7)) \ll \text{avgShift}) - ((\text{sumRef} * \text{sumRef}) \gg \text{precShift}) \quad (\text{I-222})$$

$$\text{numerDiv} = \text{Clip3}(0, 2 * \text{denomDiv}, ((\text{sumProdRefCur} + (\text{sumRefSquare} \gg 7)) \ll \text{avgShift}) - ((\text{sumRef} * \text{sumCur}) \gg \text{precShift})) \quad (\text{I-223})$$
 - The variables `shiftNumer` and `shiftDenom` are derived as follows:

$$\text{shiftDenom} = \text{Max}(0, \text{Floor}(\text{Log2}(\text{Abs}(\text{denomDiv}))) - 5) \quad (\text{I-224})$$

$$\text{shiftNumer} = \text{Max}(0, \text{shiftDenom} - 12) \quad (\text{I-225})$$
 - The variables `sNumerDiv` and `sDenomDiv` are derived as follows:

$$\text{sDenomDiv} = \text{denomDiv} \gg \text{shiftDenom} \quad (\text{I-226})$$

$$sNumerDiv = numerDiv \gg shiftNumer \quad (I-227)$$

- The value of variable divCoeff is derived from Table I.3 depending on the value of sDenomDiv, and the variables icWeight and icOffset are derived as follows:

$$icWeight = (sNumerDiv * divCoeff) \gg (shiftDenom - shiftNumer + 10) \quad (I-228)$$

$$icOffset = (sumCur - ((icWeight * sumRef) \gg 5) + avgOffset) \gg avgShift \quad (I-229)$$

- Otherwise (cIdx is not equal to 0), the following applies:

$$icWeight = 32 \quad (I-230)$$

$$icOffset = (sumCur - sumRef + avgOffset) \gg avgShift \quad (I-231)$$

Table I.3 – Specification of divCoeff depending on sDenomDiv

sDenomDiv	0	1	2	3	4	5	6	7	8	9	10	11	12
divCoeff	0	32768	16384	10923	8192	6554	5461	4681	4096	3641	3277	2979	2731
sDenomDiv	13	14	15	16	17	18	19	20	21	22	23	24	25
divCoeff	2521	2341	2185	2048	1928	1820	1725	1638	1560	1489	1425	1365	1311
sDenomDiv	26	27	28	29	30	31	32	33	34	35	36	37	38
divCoeff	1260	1214	1170	1130	1092	1057	1024	993	964	936	910	886	862
sDenomDiv	39	40	41	42	43	44	45	46	47	48	49	50	51
divCoeff	840	819	799	780	762	745	728	712	697	683	669	655	643
sDenomDiv	52	53	54	55	56	57	58	59	60	61	62	63	
divCoeff	630	618	607	596	585	575	565	555	546	537	529	520	

I.8.5.3.3.3 Bilinear sample interpolation and residual prediction process

I.8.5.3.3.3.1 General

The process is only invoked when iv_res_pred_weight_idx[xCb][yCb] is not equal to 0.

Inputs to this process are:

- a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a luma location (xBl, yBl) specifying the top-left sample of the current luma prediction block relative to the top-left sample of the current luma coding block,
- a variable nCbS specifying the size of the current luma coding block,
- two variables nPbW and nPbH specifying the width and the height of the current luma prediction block,
- two prediction list utilization flags predFlagL0 and predFlagL1,
- two reference indices refIdxL0 and refIdxL1,
- two motion vectors mvL0 and mvL1,
- when ChromaArrayType is not equal to 0, two motion vectors mvCL0 and mvCL1,
- a prediction list indication X.

Outputs of this process are:

- the (nPbW)x(nPbH) array predSamplesLXL,
- when ChromaArrayType is not equal to 0, the (nPbW / SubWidthC)x(nPbH / SubHeightC) arrays predSamplesLXCb and predSamplesLXCcr.

The location (xPb, yPb) is derived as follows:

$$xPb = xCb + xBl \quad (I-232)$$

$$yPb = yCb + yBl \quad (I-233)$$

The prediction list indication variable Y is set equal to (1 - X) and the variable availFlag is set equal to 0.

The variable `ivRefFlagLX` is set equal to $(\text{DiffPicOrderCnt}(\text{CurrPic}, \text{RefPicListX}[\text{refIdxLX}]) == 0)$ and the variable `ivRefFlagLY` is set equal to $\text{predFlagLY} ? (\text{DiffPicOrderCnt}(\text{CurrPic}, \text{RefPicListY}[\text{refIdxLY}]) == 0) : 0$.

Depending on the values of `ivRefFlagLX`, `ivRefFlagLY`, and `RpRefIdxLX`, the following applies:

- If `ivRefFlagLX` is equal to 0, the variable `availFlag` is set equal to 1, the variable `refIdxLX` is set equal to `RpRefIdxLX`, and the residual prediction motion vector scaling process as specified in clause I.8.5.3.3.3.4 is invoked with the prediction list indication variable equal to X, the motion vector `mvLX`, and the picture `RefPicListX[refIdxLX]` as inputs, and the modified motion vector `mvLX` as output.
- Otherwise (when `ivRefFlagLX` is equal to 1), the following applies:
 - If `predFlagLY` is equal to 1 and `ivRefFlagLY` is equal to 0, the following applies:
 - The variable `availFlag` is set equal to 1.
 - The residual prediction motion vector scaling process as specified in clause I.8.5.3.3.3.4 is invoked with the prediction list indication variable equal to Y, the motion vector `mvLY`, and the picture `RefPicListY[refIdxLY]` as inputs, and the modified motion vector `mvLY` as output.
 - The motion vector `mvT` is set equal to `mvLY` and the prediction list indication variable Z is set equal to Y.
 - Otherwise (`predFlagLY` is equal to 0 or `ivRefFlagLY` is equal to 1), the following applies:
 - The variable W is set equal to $(\text{predFlagLY} \ \&\& \ \text{ivRefFlagLY}) ? 0 : X$.
 - The derivation process for a motion vector from a reference block for residual prediction as specified in clause I.8.5.3.3.3.5 is invoked with the luma location (xPb, yPb) , the variables `nPbW` and `nPbH`, the picture `RefPicListW[refIdxLW]`, and the motion vector `mvLW` as inputs, and the flag `availFlag`, the motion vector `mvT`, and the prediction list utilization variable Z as outputs.
 - When `availFlag` is equal to 0 and `RpRefPicAvailFlagLW` is equal to 1, `availFlag` is set equal to 1, `mvT` is set equal to $(0, 0)$, and Z is set equal to W.

When `ChromaArrayType` is not equal to 0, the derivation process for chroma motion vectors in clause 8.5.3.2.10 is invoked with `mvLX` as input, and the output being `mvCLX`.

The array `predSamplesLXL`, and, when `ChromaArrayType` is not equal to 0, the arrays `predSamplesLXCb` and `predSamplesLXCr` are derived as follows:

- The reference picture consisting of an ordered two-dimensional array `refPicLXL` of luma samples and, when `ChromaArrayType` is not equal to 0, two ordered two-dimensional arrays `refPicLXCb` and `refPicLXCr` of chroma samples, are derived by invoking the reference picture selection process as specified in clause 8.5.3.3.2 with `refIdxLX` as input.
- The array `predSamplesLXL` and, when `ChromaArrayType` is not equal to 0, the arrays `predSamplesLXCb` and `predSamplesLXCr`, are derived by invoking the bilinear sample interpolation process specified in clause I.8.5.3.3.3.2 with the luma locations (xCb, yCb) and (xBl, yBl) , the luma prediction block width `nPbW`, the luma prediction block height `nPbH`, the motion vector `mvLX`, and, when `ChromaArrayType` is not equal to 0, the motion vector `mvCLX`, the reference array `refPicLXL` and, when `ChromaArrayType` is not equal to 0, the arrays `refPicLXCb` and `refPicLXCr` as inputs.

When one of the following conditions is true, `availFlag` is set equal to 0:

- `ivRefFlagLX` is equal to 0 and `RefRpRefAvailFlagLX[RefViewIdx[xPb][yPb]]` is equal to 0.
- `ivRefFlagLX` is equal to 1 and `RefRpRefAvailFlagLZ[ViewIdx(RefPicListX[refIdxLX])]` is equal to 0.

When `availFlag` is equal to 1, the following applies:

- Depending on the value of `ivRefFlagLX`, the variables `rpPic`, `rpRefPic`, and `mvRp` are derived as follows:
 - If `ivRefFlagLX` is equal to 0, the following applies:
 - Let `rpPic` be the picture with `PicOrderCnt(rpPic)` equal to `PicOrderCntVal` and `nuh_layer_id` equal to `ViewCompLayerId[RefViewIdx[xPb][yPb]][DepthFlag]`.
 - Let `rpRefPic` be the picture with `PicOrderCnt(rpRefPic)` equal to `PicOrderCnt(RefPicListX[RpRefIdxLX])` and `nuh_layer_id` equal to `ViewCompLayerId[RefViewIdx[xPb][yPb]][DepthFlag]`.
 - The variable `mvRp` is set equal to `DispVec[xPb][yPb]`.

- Otherwise (ivRefFlagLX is equal to 1), the following applies:
 - Let rpPic be the picture RefPicListZ[RpRefIdxLZ].
 - Let rpRefPic be the picture with PicOrderCnt(rpRefPic) equal to PicOrderCnt(rpPic) and nuh_layer_id equal to ViewCompLayerId[ViewIdx(RefPicListX[refIdxLX])][DepthFlag].
 - The variable mvRp is set equal to mvT.
- When ChromaArrayType is not equal to 0, the derivation process for chroma motion vectors in clause 8.5.3.2.10 is invoked with mvRp as input, and the output being mvRpC.
- The array rpSamplesLXL and, when ChromaArrayType is not equal to 0, the arrays rpSamplesLXCb and rpSamplesLXCcr are derived as follows:
 - Let the reference sample array rpPicLXL correspond to the decoded sample array SL derived in clause 8.7 for the previously decoded picture rpPic.
 - When ChromaArrayType is not equal to 0, let the reference sample arrays rpPicLXCb and rpPicLXCcr correspond to the decoded sample arrays SCb and SCr, respectively, derived in clause 8.7 for the previously decoded picture rpPic.
 - The array rpSamplesLXL and, when ChromaArrayType is not equal to 0, the arrays rpSamplesLXCb and rpSamplesLXCcr are derived by invoking the bilinear sample interpolation process specified in clause I.8.5.3.3.3.2 with the luma locations (xCb, yCb) and (xBl, yBl), the luma prediction block width nPbW, the luma prediction block height nPbH, the motion vectors mvLX equal to mvRp, and, when ChromaArrayType is not equal to 0, the motion vector mvCLX equal to mvRpC, the reference array rpPicLXL and, when ChromaArrayType is not equal to 0, the arrays rpPicLXCb and rpPicLXCcr as inputs.
- The array rpRefSamplesLXL and, when ChromaArrayType is not equal to 0, the arrays rpRefSamplesLXCb and rpRefSamplesLXCcr are derived as follows:
 - Let the reference sample array rpRefPicLXL correspond to the decoded sample array SL derived in clause 8.7 for the previously decoded picture rpRefPic.
 - When ChromaArrayType is not equal to 0, let the reference sample arrays rpRefPicLXCb and rpRefPicLXCcr correspond to the decoded sample arrays SCb and SCr, respectively, derived in clause 8.7 for the previously decoded picture rpRefPic.
 - The array rpRefSamplesLXL and, when ChromaArrayType is not equal to 0, the arrays rpRefSamplesLXCb and rpRefSamplesLXCcr are derived by invoking the bilinear sample interpolation process specified in clause I.8.5.3.3.3.2 with the luma locations (xCb, yCb) and (xBl, yBl), the luma prediction block width nPbW, the luma prediction block height nPbH, the motion vector mvLX equal to (mvLX + mvRp), and, when ChromaArrayType is not equal to 0, the motion vector mvCLX equal to (mvCLX + mvRpC), the reference arrays with rpRefPicLXL, and, when ChromaArrayType is not equal to 0, the arrays rpRefPicLXCb and rpRefPicLXCcr as inputs.
- The variable shiftVal is set equal to (iv_res_pred_weight_idx[xCb][yCb] – 1).
- The modified prediction samples predSamplesLXL[x][y] with $x = 0..(nPbW) - 1$ and $y = 0..(nPbH) - 1$ are derived as follows:

$$\text{predSamplesLXL}[x][y] = \text{predSamplesLXL}[x][y] + ((\text{rpSamplesLXL}[x][y] - \text{rpRefSamplesLXL}[x][y]) \gg \text{shiftVal}) \quad (\text{I-234})$$

- When ChromaArrayType is not equal to 0 and nPbW is greater than 8, the following applies:
 - The modified prediction samples predSamplesLXCb[x][y] with $x = 0..(nPbW / \text{SubWidthC}) - 1$ and $y = 0..(nPbH / \text{SubHeightC}) - 1$ are derived as follows:

$$\text{predSamplesLXCb}[x][y] = \text{predSamplesLXCb}[x][y] + ((\text{rpSamplesLXCb}[x][y] - \text{rpRefSamplesLXCb}[x][y]) \gg \text{shiftVal}) \quad (\text{I-235})$$

- The modified prediction samples predSamplesLXCcr[x][y] with $x = 0..(nPbW / \text{SubWidthC}) - 1$ and $y = 0..(nPbH / \text{SubHeightC}) - 1$ are derived as follows:

$$\text{predSamplesLXCcr}[x][y] = \text{predSamplesLXCcr}[x][y] + ((\text{rpSamplesLXCcr}[x][y] - \text{rpRefSamplesLXCcr}[x][y]) \gg \text{shiftVal}) \quad (\text{I-236})$$

I.8.5.3.3.2 Bilinear sample interpolation process

The specifications in clause 8.5.3.3.1 apply with the following modifications:

- All invocations of the process specified in clause 8.5.3.3.3.2 are replaced with invocations of the process specified in clause I.8.5.3.3.3.3 with chromaFlag equal to 0 as additional input.
- All invocations of the process specified in clause 8.5.3.3.3.3 are replaced with invocations of the process specified in clause I.8.5.3.3.3.3 with chromaFlag equal to 1 as additional input.

I.8.5.3.3.3.3 Bilinear luma and chroma sample interpolation process

Inputs to this process are:

- a location in full-sample units (xInt, yInt),
- a location offset in fractional-sample units (xFrac, yFrac),
- a sample reference sample array refPicLX,
- a flag chromaFlag.

Output of this process is a predicted sample value predSampleLX.

Depending on the value of chromaFlag, the following applies:

- If chromaFlag is equal 0, the following applies:
 - The variables xFrac and yFrac are set equal to (xFrac << 1) and (yFrac << 1), respectively.
 - The variables picWidth and picHeight are set equal to pic_width_in_luma_samples and pic_height_in_luma_samples, respectively.
- Otherwise (chromaFlag is equal to 1), the variables picWidth and picHeight are set equal to PicWidthInSamplesC and PicHeightInSamplesC, respectively.

In Figure 8-5, the positions labelled with upper-case letters $B_{i,j}$ within shaded blocks represent samples at full-sample locations inside the given two-dimensional array refPicLX of samples. These samples may be used for generating the predicted sample value predSampleLX. The locations ($x_{B_{i,j}}$, $y_{B_{i,j}}$) for each of the corresponding samples $B_{i,j}$ inside the given array refPicLX of samples are derived as follows:

$$x_{B_{i,j}} = \text{Clip3}(0, \text{picWidth} - 1, x_{\text{Int}} + i) \quad (\text{I-237})$$

$$y_{B_{i,j}} = \text{Clip3}(0, \text{picHeight} - 1, y_{\text{Int}} + j) \quad (\text{I-238})$$

The positions labelled with lower-case letters within un-shaded blocks represent samples at eighth-pel sample fractional locations. The location offset in fractional-sample units (xFrac, yFrac) specifies which of the generated samples at full-sample and fractional-sample locations is assigned to the predicted sample value predSampleLX. This assignment is as specified in Table 8-9, with xFracC, yFracC, and predSampleLXC replaced by xFrac, yFrac, and predSampleLX, respectively. The output is the value of predSampleLX.

The variables shift1, shift2, and shift3 are derived as follows:

- The variable bitDepth is set equal to chromaFlag ? BitDepthC : BitDepthY.
- The variable shift1 is set equal to Min(4, bitDepth – 8), the variable shift2 is set equal to 6, and the variable shift3 is set equal to Max(2, 14 – bitDepth).

Given the samples $B_{i,j}$ at full-sample locations ($x_{B_{i,j}}$, $y_{B_{i,j}}$), the samples $ab_{0,0}$ to $hh_{0,0}$ at fractional sample positions are derived as follows:

- The samples labelled $ab_{0,0}$, $ac_{0,0}$, $ad_{0,0}$, $ae_{0,0}$, $af_{0,0}$, $ag_{0,0}$, and $ah_{0,0}$ are derived by applying a 2-tap filter to the nearest integer position samples as follows:

$$ab_{0,0} = (56 * B_{0,0} + 8 * B_{1,0}) \gg \text{shift1} \quad (\text{I-239})$$

$$ac_{0,0} = (48 * B_{0,0} + 16 * B_{1,0}) \gg \text{shift1} \quad (\text{I-240})$$

$$ad_{0,0} = (40 * B_{0,0} + 24 * B_{1,0}) \gg \text{shift1} \quad (\text{I-241})$$

$$ae_{0,0} = (32 * B_{0,0} + 32 * B_{1,0}) \gg \text{shift1} \quad (\text{I-242})$$

$$af_{0,0} = (24 * B_{0,0} + 40 * B_{1,0}) \gg \text{shift1} \quad (\text{I-243})$$

$$ag_{0,0} = (16 * B_{0,0} + 48 * B_{1,0}) \gg \text{shift1} \quad (\text{I-244})$$

$$ah_{0,0} = (8 * B_{0,0} + 56 * B_{1,0}) \gg \text{shift1} \quad (\text{I-245})$$

- The samples labelled $ba_{0,0}$, $ca_{0,0}$, $da_{0,0}$, $ea_{0,0}$, $fa_{0,0}$, $ga_{0,0}$, and $ha_{0,0}$ are derived by applying a 2-tap filter to the nearest

integer position samples as follows:

$$ba_{0,0} = (56 * B_{0,0} + 8 * B_{0,1}) \gg \text{shift1} \quad (\text{I-246})$$

$$ca_{0,0} = (48 * B_{0,0} + 16 * B_{0,1}) \gg \text{shift1} \quad (\text{I-247})$$

$$da_{0,0} = (40 * B_{0,0} + 24 * B_{0,1}) \gg \text{shift1} \quad (\text{I-248})$$

$$ea_{0,0} = (32 * B_{0,0} + 32 * B_{0,1}) \gg \text{shift1} \quad (\text{I-249})$$

$$fa_{0,0} = (24 * B_{0,0} + 40 * B_{0,1}) \gg \text{shift1} \quad (\text{I-250})$$

$$ga_{0,0} = (16 * B_{0,0} + 48 * B_{0,1}) \gg \text{shift1} \quad (\text{I-251})$$

$$ha_{0,0} = (8 * B_{0,0} + 56 * B_{0,1}) \gg \text{shift1} \quad (\text{I-252})$$

- The samples labelled $bX_{0,0}$, $cX_{0,0}$, $dX_{0,0}$, $eX_{0,0}$, $fX_{0,0}$, $gX_{0,0}$, and $hX_{0,0}$ for X being replaced by b, c, d, e, f, g, and h, respectively, are derived by applying a 2-tap filter to the intermediate values $aX_{0,i}$ with $i = 0..1$ in the vertical direction as follows:

$$bX_{0,0} = (56 * aX_{0,0} + 8 * aX_{0,1}) \gg \text{shift2} \quad (\text{I-253})$$

$$cX_{0,0} = (48 * aX_{0,0} + 16 * aX_{0,1}) \gg \text{shift2} \quad (\text{I-254})$$

$$dX_{0,0} = (40 * aX_{0,0} + 24 * aX_{0,1}) \gg \text{shift2} \quad (\text{I-255})$$

$$eX_{0,0} = (32 * aX_{0,0} + 32 * aX_{0,1}) \gg \text{shift2} \quad (\text{I-256})$$

$$fX_{0,0} = (24 * aX_{0,0} + 40 * aX_{0,1}) \gg \text{shift2} \quad (\text{I-257})$$

$$gX_{0,0} = (16 * aX_{0,0} + 48 * aX_{0,1}) \gg \text{shift2} \quad (\text{I-258})$$

$$hX_{0,0} = (8 * aX_{0,0} + 56 * aX_{0,1}) \gg \text{shift2} \quad (\text{I-259})$$

I.8.5.3.3.4 Residual prediction motion vector scaling process

Inputs to this process are:

- a prediction list indication variable X,
- a motion vector mvLX,
- a reference picture (associated with the motion vector mvLX) refPicLX.

Output of this process is a scaled motion vector mvLX.

The motion vector mvLX is scaled as follows:

$$tx = (16384 + (\text{Abs}(td) \gg 1)) / td \quad (\text{I-260})$$

$$\text{distScaleFactor} = \text{Clip3}(-4096, 4095, (tb * tx + 32) \gg 6) \quad (\text{I-261})$$

$$\text{mv} = \text{Clip3}(-32768, 32767, \text{Sign}(\text{distScaleFactor} * \text{mvLX}) * ((\text{Abs}(\text{distScaleFactor} * \text{mvLX}) + 127) \gg 8)) \quad (\text{I-262})$$

where td and tb are derived as:

$$td = \text{Clip3}(-128, 127, \text{DiffPicOrderCnt}(\text{CurrPic}, \text{refPicLX})) \quad (\text{I-263})$$

$$tb = \text{Clip3}(-128, 127, \text{DiffPicOrderCnt}(\text{CurrPic}, \text{RefPicListX}[\text{RpRefIdxLX}])) \quad (\text{I-264})$$

I.8.5.3.3.5 Derivation process for a motion vector from a reference block for residual prediction

Inputs to this process are:

- a luma location (xPb, yPb) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,
- two variables nPbW and nPbH specifying the width and the height of the current luma prediction block,
- a reference picture refPic,
- a disparity vector dispVec.

Outputs of this process are:

- a flag availFlag,

- a motion vector mvT ,
- a prediction list indication variable Y .

The flag $availFlag$ is set equal to 0, the motion vector mvT is set equal to (0, 0), and the prediction list indication variable Y is set equal to 0.

The reference luma location ($xRef$, $yRef$) in $refPic$ is derived as follows:

$$xRefFull = xPb + (nPbW \gg 1) + ((dispVec[0] + 2) \gg 2) \quad (I-265)$$

$$yRefFull = yPb + (nPbH \gg 1) + ((dispVec[1] + 2) \gg 2) \quad (I-266)$$

$$xRef = Clip3(0, pic_width_in_luma_samples - 1, (xRefFull \gg 3) \ll 3) \quad (I-267)$$

$$yRef = Clip3(0, pic_height_in_luma_samples - 1, (yRefFull \gg 3) \ll 3) \quad (I-268)$$

Let $refPb$ be the luma prediction block covering the luma position ($xRef$, $yRef$) in the picture $refPic$.

The variable $cuPredModeRef[x][y]$ is set equal to $CuPredMode[x][y]$ of the picture $refPic$.

When $cuPredModeRef[xRef][yRef]$ is equal to $MODE_SKIP$ or $MODE_INTER$, the following applies for X in the range of 0 to 1, inclusive:

- The variables $predFlagRef[x][y]$, $mvRef[x][y]$, and $refIdxRef[x][y]$ are set equal to $PredFlagLX[x][y]$, $MvLX[x][y]$, and $RefIdxLX[x][y]$, respectively, of picture $refPic$.
- The variable $refPicListRef$ is set equal to $RefPicListX$ of the slice containing $refPb$ in the picture $refPic$.
- When $availFlag$ is equal to 0, $predFlagRef[xRef][yRef]$ is equal to 1, $DiffPicOrderCnt(refPic, refPicListRef[refIdxRef[xRef][xRef]])$ is not equal to 0, and $RpRefPicAvailFlagLX$ is equal to 1, the following applies:
 - The variable $availFlag$ is set equal to 1.
 - The variable Y is set equal to X .
 - The residual prediction motion vector scaling process as specified in clause I.8.5.3.3.4 is invoked with the prediction list indication variable equal to X , the motion vector $mvRef[xRef][yRef]$, and the reference picture $refPicListRef[refIdxRef[xRef][xRef]]$ as inputs, and the output being the motion vector mvT .

I.8.5.3.4 Decoding process for inter sample prediction for rectangular sub-block partitions

Inputs to this process are:

- a luma location (xCb , yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a luma location (xBI , yBI) specifying the top-left sample of the current luma prediction block relative to the top-left sample of the current luma coding block,
- a variable $nCbS$ specifying the size of the current luma coding block,
- two variables $nPbW$ and $nPbH$ specifying the width and the height of the luma prediction block.

Outputs of this process are:

- an $(nCbSL) \times (nCbSL)$ array $predSamples_L$ of luma prediction samples, where $nCbSL$ is derived as specified below,
- when $ChromaArrayType$ is not equal to 0, an $(nCbSw_C) \times (nCbSh_C)$ array $predSamples_{Cb}$ of chroma prediction samples for the component Cb , where $nCbSw_C$ and $nCbSh_C$ are derived as specified below,
- when $ChromaArrayType$ is not equal to 0, an $(nCbSw_C) \times (nCbSh_C)$ array $predSamples_{Cr}$ of chroma prediction samples for the component Cr , where $nCbSw_C$ and $nCbSh_C$ are derived as specified below.

When $ChromaArrayType$ is not equal to 0, the variable $nCbSw_C$ is set equal to ($nCbS / SubWidthC$) and the variable $nCbSh_C$ is set equal to ($nCbS / SubHeightC$).

The luma location (xPb , yPb) specifying the top-left sample of the current luma prediction block relative to the top-left sample of the current picture is set equal to ($xCb + xBI$, $yCb + yBI$).

The variables $nSbW$ and $nSbH$ specifying the width and height of the sub-block partitions are derived as follows:

$$(nSbW, nSbH) = SubPbArrayPartSize[xPb][yPb] \quad (I-269)$$

For x in the range of 0 to ($nPbW / nSbW - 1$), inclusive, the following applies:

- For y in the range of 0 to $(n_{PbH} / n_{SbH} - 1)$, inclusive, the following applies:
 - The luma location (x_{Sb}, y_{Sb}) specifying the top-left sample of the current luma sub-block partition relative to the top-left sample of the current luma coding block is derived as follows:

$$x_{Sb} = x_{Bl} + x * n_{SbW} \quad (I-270)$$

$$y_{Sb} = y_{Bl} + y * n_{SbH} \quad (I-271)$$
 - For X in the range of 0 to 1, inclusive, the variables $mvLX$, $mvCLX$, $refIdxLX$, and $predFlagLX$ are derived as follows:

$$mvLX = \text{SubPbArrayMvLX}[x_{Cb} + x_{Sb}][y_{Cb} + y_{Sb}] \quad (I-272)$$

$$mvCLX = (\text{ChromaArrayType} \neq 0) ? \text{SubPbArrayMvCLX}[x_{Cb} + x_{Sb}][y_{Cb} + y_{Sb}] : (0, 0) \quad (I-273)$$

$$refIdxLX = \text{SubPbArrayRefIdxLX}[x_{Cb} + x_{Sb}][y_{Cb} + y_{Sb}] \quad (I-274)$$

$$predFlagLX = \text{SubPbArrayPredFlagLX}[x_{Cb} + x_{Sb}][y_{Cb} + y_{Sb}] \quad (I-275)$$
 - The decoding process for inter sample prediction as specified in clause I.8.5.3.3.1 is invoked with the luma coding block location (x_{Cb}, y_{Cb}) , the luma prediction block location (x_{Bl}, y_{Bl}) equal to (x_{Sb}, y_{Sb}) , the luma coding block size n_{CbS} , the luma prediction block width n_{PbW} equal to n_{SbW} , the luma prediction block height n_{PbH} equal to n_{SbH} , the luma motion vectors $mvL0$ and $mvL1$, when ChromaArrayType is not equal to 0, the chroma motion vectors $mvCL0$ and $mvCL1$, the reference indices $refIdxL0$ and $refIdxL1$, and the prediction list utilization flags $predFlagL0$ and $predFlagL1$ as inputs, and the inter prediction samples that are an $(n_{CbSL}) \times (n_{CbSL})$ array $predSamples_L$ of prediction luma samples, and, when ChromaArrayType is not equal to 0, two $(n_{CbSwC}) \times (n_{CbShC})$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$ of prediction chroma samples, one for each of the chroma components Cb and Cr , as outputs.

I.8.5.3.5 Decoding process for inter sample prediction for depth predicted sub-block partitions

I.8.5.3.5.1.1 General

Inputs to this process are:

- a luma location (x_{Cb}, y_{Cb}) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a variable n_{CbS} specifying the size of the current luma coding block,
- two luma motion vectors $mvL0$ and $mvL1$,
- when ChromaArrayType is not equal to 0, the chroma motion vectors $mvCL0$ and $mvCL1$,
- two reference indices $refIdxL0$ and $refIdxL1$,
- two prediction list utilization flags $predFlagL0$, and $predFlagL1$,
- a variable $partIdx$ specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

- an $(n_{CbSL}) \times (n_{CbSL})$ array $predSamples_L$ of luma prediction samples, where n_{CbSL} is derived as specified below,
- when ChromaArrayType is not equal to 0, an $(n_{CbSwC}) \times (n_{CbShC})$ array $predSamples_{Cb}$ of chroma prediction samples for the component Cb , where n_{CbSwC} and n_{CbShC} are derived as specified below,
- when ChromaArrayType is not equal to 0, an $(n_{CbSwC}) \times (n_{CbShC})$ array $predSamples_{Cr}$ of chroma prediction samples for the component Cr , where n_{CbSwC} and n_{CbShC} are derived as specified below.

The variable n_{CbSL} is set equal to n_{CbS} . When ChromaArrayType is not equal to 0, the variable n_{CbSwC} is set equal to $n_{CbS} / \text{SubWidthC}$ and the variable n_{CbShC} is set equal to $n_{CbS} / \text{SubHeightC}$.

The decoding process for inter sample prediction as specified in clause I.8.5.3.3.1 is invoked with the luma coding block location (x_{Cb}, y_{Cb}) , the luma prediction block location (x_{Bl}, y_{Bl}) set to $(0, 0)$, the luma coding block size n_{CbS} , the luma prediction block width n_{PbW} equal to n_{CbS} , the luma prediction block height n_{PbH} equal to n_{CbS} , the luma motion vectors $mvL0$ and $mvL1$, when ChromaArrayType is not equal to 0, the chroma motion vectors $mvCL0$ and $mvCL1$, the reference indices $refIdxL0$ and $refIdxL1$, and the prediction list utilization flags $predFlagL0$ and $predFlagL1$ as inputs, and the inter prediction samples ($predSamples$) that are an $(n_{CbSL}) \times (n_{CbSL})$ array $predSamples_L$ of prediction luma samples and, when ChromaArrayType is not equal to 0, two $(n_{CbSwC}) \times (n_{CbShC})$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$ of prediction chroma samples, one for each of the chroma components Cb and Cr , as outputs.

The partition pattern contourPattern is derived as follows:

- The derivation process for a depth or disparity sample array from a depth picture as specified in clause I.8.5.7 is invoked with the luma location (xBlk, yBlk) equal to (xCb, yCb), the variable nBlkW equal to nCbS, the variable nBlkH equal to nCbS, the disparity vector dispVec equal to DispRefVec[xCb][yCb], the reference view order index refViewIdx equal to RefViewIdx[xCb][yCb], and the variable partIdx equal to 1 as inputs, and the output is the array refSamples of size (nCbS)x(nCbS).

- The variable threshVal is derived as follows:

$$\text{threshVal} = (\text{refSamples}[0][0] + \text{refSamples}[0][\text{nCbS} - 1] + \text{refSamples}[\text{nCbS} - 1][0] + \text{refSamples}[\text{nCbS} - 1][\text{nCbS} - 1]) \gg 2 \quad (\text{I-276})$$

- For $x = 0.. \text{nCbS} - 1$ and $y = 0.. \text{nCbS} - 1$, the following applies:

$$\text{contourPattern}[x][y] = (\text{refSamples}[x][y] > \text{threshVal}) \quad (\text{I-277})$$

The array TempSamplesDbbp_L and, when ChromaArrayType is not equal to 0, the arrays TempSamplesDbbp_{Cb} and TempSamplesDbbp_{Cr} are modified as follows:

```
for( y = 0; y < nCbSL; y++ )
  for( x = 0; x < nCbSL; x++ )
    if( contourPattern[ x ][ y ] == ( partIdx != contourPattern[ 0 ][ 0 ] ) ) {
      TempSamplesDbbpL[ x ][ y ] = predSamplesL[ x ][ y ]
      if( ChromaArrayType != 0 && ( x % SubWidthC ) == 0 && ( y % SubHeightC ) == 0 ) {
        xC = x / SubWidthC
        yC = y / SubHeightC
        TempSamplesDbbpCb[ xC ][ yC ] = predSamplesCb[ xC ][ yC ]
        TempSamplesDbbpCr[ xC ][ yC ] = predSamplesCr[ xC ][ yC ]
      }
    }
```

(I-278)

When partIdx is equal to 1, the array predSamples_L and, when ChromaArrayType is not equal to 0, the arrays predSamples_{Cb} and predSamples_{Cr} are modified as follows:

- The derivation process for contour boundary filtered samples as specified in clause I.8.5.3.5.1.2 is invoked with, the luma coding block size nCbS_L, the current coding block width nCbSw equal to nCbS_L, the current coding block height nCbSh equal to nCbS_L, the partition pattern contourPattern, and the array predSamples of prediction samples equal to TempSamplesDbbp_L as inputs, and the output is assigned to the array predSamples_L of luma prediction samples.
- When ChromaArrayType is not equal to 0, the derivation process for contour boundary filtered samples as specified in clause I.8.5.3.5.1.2 is invoked with, the luma coding block size nCbS_L, the current coding block width nCbSw equal to nCbSw_C, the current coding block height nCbSh equal to nCbSh_C, the partition pattern contourPattern, and the array predSamples of prediction samples equal to TempSamplesDbbp_{Cb} as inputs, and the output is assigned to the array predSamples_{Cb} of chroma prediction samples.
- When ChromaArrayType is not equal to 0, the derivation process for contour boundary filtered samples as specified in clause I.8.5.3.5.1.2 is invoked with, the luma coding block size nCbS_L, the current coding block width nCbSw equal to nCbSw, the current coding block height nCbSh equal to nCbSh_C, the partition pattern contourPattern, and the array predSamples of prediction samples equal to TempSamplesDbbp_{Cr} as inputs, and the output is assigned to the array predSamples_{Cr} of chroma prediction samples.

I.8.5.3.5.1.2 Derivation process for contour boundary filtered samples

Inputs to this process are:

- a variable nCbS_L specifying the size of the current luma coding block,
- a variable nCbSw specifying the width of the current luma or chroma coding block,
- a variable nCbSh specifying the height of the current luma or chroma coding block,
- an (nCbS_L)x(nCbS_L) partition pattern contourPattern,
- an (nCbSw)x(nCbSh) array predSamples prediction samples.

Output of this process is a modified (nCbSw)x(nCbSh) array predSamples of prediction samples.

The (nCbSw)x(nCbSh) array p is set equal to predSamples.

The variable xOff, yOff, nCbS_N, and n are derived as follows:

- If PartMode is equal to PART_Nx2N, xOff is set equal to 1, yOff is set equal to 0, nCbS_N is set equal to nCbSw, and n is set equal to (nCbS_L / nCbSw).
- Otherwise (PartMode is not equal to PART_Nx2N), xOff is set equal to 0, yOff is set equal to 1, nCbS_N is set equal to nCbSh, and n is set equal to (nCbS_L / nCbSh).

The values of predSamples are derived as follows:

```

for( y = 0; y < nCbSh; y++ )
  for( x = 0; x < nCbSw; x++ ) {
    filt = p[ x ][ y ]
    prevFlag = contourPattern[ Max( 0, n * ( x - xOff ) ) ][ Max( 0, n * ( y - yOff ) ) ]
    nextFlag = contourPattern[ Min( n * ( x + xOff ), nCbSL - 1 ) ][ Min( n * ( y + yOff ), nCbSL - 1 ) ]
    if( prevFlag != nextFlag )
      filt = ( p[ Max( 0, x - xOff ) ][ Max( 0, y - yOff ) ] + ( filt << 1 ) +
        p[ Min( x + xOff, nCbSN - 1 ) ][ Min( y + yOff, nCbSN - 1 ) ] ) >> 2
    predSamples[ x ][ y ] = filt
  }

```

(I-279)

I.8.5.4 Decoding process for the residual signal of coding units coded in inter prediction mode

Inputs to this process are:

- a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a variable log2CbSize specifying the size of the current luma coding block.

Outputs of this process are:

- an (nCbS_L)x(nCbS_L) array resSamples_L of luma residual samples, where nCbS_L is derived as specified below,
- when ChromaArrayType is not equal to 0, an (nCbSw_C)x(nCbSh_C) array resSamples_{Cb} of chroma residual samples for the component Cb, where nCbSw_C and nCbSh_C are derived as specified below,
- when ChromaArrayType is not equal to 0, an (nCbSw_C)x(nCbSh_C) array resSamples_{Cr} of chroma residual samples for the component Cr, where nCbSw_C and nCbSh_C are derived as specified below.

The variable nCbS_L is set equal to 1 << log2CbSize. When ChromaArrayType is not equal to 0, the variable nCbSw_C is set equal to nCbS_L / SubWidthC and the variable nCbSh_C is set equal to nCbS_L / SubHeightC.

Let resSamples_L be an (nCbS_L)x(nCbS_L) array of luma residual samples and, when ChromaArrayType is not equal to 0, let resSamples_{Cb} and resSamples_{Cr} be two (nCbSw_C)x(nCbSh_C) arrays of chroma residual samples.

- If DcOnlyFlag[xCb][yCb] is equal to 0, the following applies, depending on the value of rqt_root_cbf:
 - If rqt_root_cbf is equal to 0 or cu_skip_flag[xCb][yCb] is equal to 1, all samples of the (nCbS_L)x(nCbS_L) array resSamples_L and, when ChromaArrayType is not equal to 0, all samples of the two (nCbSw_C)x(nCbSh_C) arrays resSamples_{Cb} and resSamples_{Cr} are set equal to 0.
 - Otherwise (rqt_root_cbf is equal to 1), the following ordered steps apply:
 1. The decoding process for luma residual blocks as specified in clause 8.5.4.2 is invoked with the luma location (xCb, yCb), the luma location (xB0, yB0) set equal to (0, 0), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable nCbS set equal to nCbS_L, and the (nCbS_L)x(nCbS_L) array resSamples_L as inputs, and a modified version of the (nCbS_L)x(nCbS_L) array resSamples_L as output.
 2. When ChromaArrayType is not equal to 0, the decoding process for chroma residual blocks as specified in clause 8.5.4.3 is invoked with the luma location (xCb, yCb), the luma location (xB0, yB0) set equal to (0, 0), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable cIdx set equal to 1, the variable nCbSw set equal to nCbSw_C, the variable nCbSh set equal to nCbSh_C, and the (nCbSw_C)x(nCbSh_C) array resSamples_{Cb} as inputs, and a modified version of the (nCbSw_C)x(nCbSh_C) array resSamples_{Cb} as output.
 3. When ChromaArrayType is not equal to 0, the decoding process for chroma residual blocks as specified in clause 8.5.4.3 is invoked with the luma location (xCb, yCb), the luma location (xB0, yB0) set equal to (0, 0), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable cIdx set equal to 2, the variable nCbSw set equal to nCbSw_C, the variable nCbSh set equal to nCbSh_C, and the (nCbSw_C)x(nCbSh_C) array resSamples_{Cr} as inputs, and a modified version of the (nCbSw_C)x(nCbSh_C) array resSamples_{Cr} as output.

- Otherwise ($\text{DcOnlyFlag}[x_{Cb}][y_{Cb}]$ is equal to 1), for x in the range of 0 to $\text{nCbS}_L - 1$, inclusive, and y in the range of 0 to $\text{nCbS}_L - 1$, inclusive, $\text{resSamples}_L[x][y]$ is set equal to $\text{DcOffset}[x_{Cb}][y_{Cb}][0]$.

NOTE – When $\text{DcOnlyFlag}[x_{Cb}][y_{Cb}]$ is equal to 1, ChromaArrayType is equal to 0 in this version of this Specification.

1.8.5.5 Derivation process for a disparity vector for texture layers

1.8.5.5.1 General

Inputs to this process are:

- a luma location (x_{Cb}, y_{Cb}) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a variable nCbS specifying the size of the current luma coding block.

The flag dvAvailFlag is set equal to 0 and the disparity vector dispVec is set equal to (0, 0).

The variable $\text{checkParallelMergeFlag}$ is derived as follows:

- If one or more of the following conditions are true, $\text{checkParallelMergeFlag}$ is set equal to 1:
 - $\text{CuPredMode}[x_{Cb}][y_{Cb}]$ is equal to MODE_SKIP .
 - $\text{CuPredMode}[x_{Cb}][y_{Cb}]$ is equal to MODE_INTER and $\text{merge_flag}[x_{Cb}][y_{Cb}]$ is equal to 1.
- Otherwise, $\text{checkParallelMergeFlag}$ is set equal to 0.

When $\text{slice_temporal_mvp_enabled_flag}$ is equal to 1, the derivation process for a disparity vector from temporal neighbouring blocks as specified in clause 1.8.5.5.2 is invoked with the luma location (x_{Cb}, y_{Cb}), and the variable nCbS as inputs, and the outputs are the flag dvAvailFlag , the disparity vector dispVec , and the reference view order index refViewIdx .

When dvAvailFlag is equal to 0, the following applies for i in the range of 0 to 1, inclusive:

1. The variable N is set equal to $(i == 0) ? A_1 : B_1$.
2. The variable (x_N, y_N) is set equal to $(i == 0) ? (x_{Cb} - 1, y_{Cb} + \text{nCbS} - 1) : (x_{Cb} + \text{nCbS} - 1, y_{Cb} - 1)$.
3. The derivation process for z-scan order block availability as specified in clause 6.4.1 is invoked with (x_{Curr}, y_{Curr}) set equal to the (x_{Cb}, y_{Cb}) and the luma location (x_N, y_N) as inputs, and the output is assigned to nbAvailFlag .
4. When $\text{CuPredMode}[x_N][y_N]$ is equal to MODE_INTRA , nbAvailFlag is set equal to 0.
5. When all of the following conditions are true, nbAvailFlag is set equal to 0.
 - $\text{checkParallelMergeFlag}$ is equal to 1,
 - $(x_{Cb} \gg \text{Log2ParMrgLevel})$ is equal to $(x_N \gg \text{Log2ParMrgLevel})$,
 - $(y_{Cb} \gg \text{Log2ParMrgLevel})$ is equal to $(y_N \gg \text{Log2ParMrgLevel})$.
6. The flag tPredNbAvailFlag is set equal to nbAvailFlag .
7. When N is equal to B_1 and $((y_N \gg \text{CtbLog2SizeY}) \ll \text{CtbLog2SizeY})$ is less than $((y_{Cb} \gg \text{CtbLog2SizeY}) \ll \text{CtbLog2SizeY})$, tPredNbAvailFlag is set equal to 0.
8. The flag $\text{tPredNbDvAvailFlagN}$ is set equal to 0.
9. For X in the range of 0 to 1, inclusive, the following applies:
 - When dvAvailFlag is equal to 0, nbAvailFlag is equal to 1, and $\text{PredFlagLX}[x_N][y_N]$ is equal to 1, the following applies:
 - If $\text{DiffPicOrderCnt}(\text{RefPicListX}[\text{RefIdxLX}[x_N][y_N]], \text{CurrPic})$ is equal to 0, the following applies:

$$\text{refViewIdx} = \text{ViewIdx}(\text{RefPicListX}[\text{RefIdxLX}[x_N][y_N]]) \quad (\text{I-280})$$

$$\text{dispVec} = \text{MvLXN}[x_N][y_N] \quad (\text{I-281})$$

$$\text{dvAvailFlag} = 1 \quad (\text{I-282})$$
 - Otherwise ($\text{DiffPicOrderCnt}(\text{RefPicListX}[\text{RefIdxLX}[x_N][y_N]], \text{CurrPic})$ is not equal to 0), when tPredNbAvailFlag is equal to 1, $\text{tPredNbDvAvailFlagN}$ is equal to 0, $\text{CuPredMode}[x_N][y_N]$ is equal to MODE_SKIP , and $\text{lvMcFlag}[x_N][y_N]$ is equal to 1, the following applies:

$$tPredNbDispVecN = DispRefVec[xN][yN] \quad (I-283)$$

$$tPredNbRefViewIdxN = RefViewIdx[xN][yN] \quad (I-284)$$

$$tPredNbDvAvailFlagN = 1 \quad (I-285)$$

For i in the range of 0 to 1, inclusive, the following applies:

- The variable N is set equal to $(i == 0) ? A_1 : B_1$.
- When $dvAvailFlag$ is equal to 0 and $tPredNbDvAvailFlagN$ is equal to 1, the following applies:

$$dispVec = tPredNbDispVecN \quad (I-286)$$

$$refViewIdx = tPredNbRefViewIdxN \quad (I-287)$$

$$dvAvailFlag = 1 \quad (I-288)$$

When $dvAvailFlag$ is equal to 0, $refViewIdx$ is set equal to $DefaultRefViewIdx$ and $dispVec$ is set equal to $(0, 0)$.

Depending on the value of $DepthRefEnabledFlag$, the following applies:

- If $DepthRefEnabledFlag$ is equal to 1, the following applies:
 - The derivation process for a depth or disparity sample array from a depth picture as specified in clause I.8.5.7 is invoked with the luma location $(xBlk, yBlk)$ equal to (xCb, yCb) , the variable $nBlkW$ equal to $nCbS$, the variable $nBlkH$ equal to $nCbS$, the disparity vector $dispVec$, the reference view order index $refViewIdx$, and the variable $partIdx$ equal to 0 as inputs, and the array $disparitySamples$ of size $(nCbS) \times (nCbS)$ as output.
 - The disparity vector $dispRefVec$ is set equal to $(disparitySamples[0][0], 0)$.
- Otherwise ($DepthRefEnabledFlag$ is equal to 0), the disparity vector $dispRefVec$ is set equal to $dispVec$.

For use in derivation processes of variables invoked later in the decoding process, the following assignments are made for $x = xCb.(xCb + nCbS - 1)$, $y = yCb.(yCb + nCbS - 1)$:

$$DispVec[x][y] = dispVec \quad (I-289)$$

$$DispRefVec[x][y] = dispRefVec \quad (I-290)$$

$$RefViewIdx[x][y] = refViewIdx \quad (I-291)$$

I.8.5.5.2 Derivation process for a disparity vector from temporal neighbouring blocks

Inputs to this process are:

- a luma location (xCb, yCb) specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a variable $nCbS$ specifying the size of the current luma coding block.

Outputs of this process are:

- the availability flag $dvAvailFlag$,
- the disparity vector $dispVec$,
- the reference view order index $refViewIdx$.

The luma location $(xRef, yRef)$ is derived as follows:

$$xRef = ((xCb + nCbS / 2) >> 4) << 4 \quad (I-292)$$

$$yRef = ((yCb + nCbS / 2) >> 4) << 4 \quad (I-293)$$

The flag $dvAvailFlag$ is set equal to 0 and $dispVec$ is set equal to $(0, 0)$.

For i from 0 to $NumDdvCandPics - 1$, inclusive, the following ordered steps apply:

1. Let $colPb$ the luma prediction block covering the luma location $(xRef, yRef)$ in picture $DdvCandPicList[i]$.
2. For X in the range of 0 to 1, inclusive, the following applies:
 - The variables $candPredFlag[x][y]$, $candRefIdx[x][y]$, and $candMV[x][y]$ are set equal to the variables $PredFlagLX[x][y]$, $RefIdxLX[x][y]$, and $MvLX[x][y]$ of the picture $DdvCandPicList[i]$, respectively.

- The variable `candPicRefPicList` is set equal to `RefPicListX` of the slice containing `colPb` in the picture `DdvCandPicList[i]`.
- When `colPb` is not coded in an intra prediction mode and `candPredFlag[xRef][yRef]` is equal to 1, the following applies:
 - The variable `candRefViewIdx` is set equal to `ViewIdx(candPicRefPicList[candRefIdx[xRef][yRef]])`.
 - When `dvAvailFlag` is equal to 0, `candRefViewIdx` is not equal to the `ViewIdx(DdvCandPicList[i])`, and there is a reference picture with `ViewIdx` equal to `candRefViewIdx` in `RefPicList0` or `RefPicList1`, the following applies:

$$\text{refViewIdx} = \text{candRefViewIdx} \quad (\text{I-294})$$

$$\text{dispVec} = \text{candMV}[\text{xRef}][\text{yRef}] \quad (\text{I-295})$$

$$\text{dvAvailFlag} = 1 \quad (\text{I-296})$$

1.8.5.6 Derivation process for a disparity vector for depth layers

Inputs to this process are:

- a luma location (`xCb, yCb`) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,
- a variable `nCbS` specifying the size of the current luma coding block.

When `DispAvailFlag` is equal to 1, the following assignments are made for `x = xCb..(xCb + nCbS – 1)` and `y = yCb..(yCb + nCbS – 1)`:

$$\text{DispVec}[x][y] = (\text{DepthToDisparityB}[\text{DefaultRefViewIdx}][1 \ll (\text{BitDepth}_Y - 1)], 0) \quad (\text{I-297})$$

$$\text{DispRefVec}[x][y] = \text{DispVec}[x][y] \quad (\text{I-298})$$

$$\text{RefViewIdx}[x][y] = \text{DefaultRefViewIdx} \quad (\text{I-299})$$

1.8.5.7 Derivation process for a depth or disparity sample array from a depth picture

Inputs to this process are:

- a luma location (`xBlk, yBlk`) specifying the top-left sample of the current luma block relative to the top-left luma sample of the current picture,
- two variables `nBlkW` and `nBlkH` specifying the width and the height of the luma block, respectively,
- a disparity vector `dispVec`,
- a reference view order index `refViewIdx`,
- a variable `partIdx` specifying a sub-block partitioning.

Outputs of this process are:

- an $(nBlkW) \times (nBlkH)$ array `dSamples` of depth or disparity values (depending on `partIdx`),
- a flag `horSplitFlag` (when `partIdx` is equal to 2).

Let `refDepPic` be the picture in the current access unit with `nuh_layer_id` equal to `ViewCompLayerId[refViewIdx][1]`.

Let `refDepPels` be the array of reconstructed luma samples S_L of the picture `refDepPic`. The luma location (`xRefBlk, yRefBlk`) of top-left luma sample of a block in `refDepPels` is derived as follows:

$$\text{xRefBlk} = \text{xBlk} + ((\text{dispVec}[0] + 2) \gg 2) \quad (\text{I-300})$$

$$\text{yRefBlk} = \text{yBlk} + ((\text{dispVec}[1] + 2) \gg 2) \quad (\text{I-301})$$

Depending on the value of `partIdx`, the variables `nSubBlkW`, `nSubBlkH`, and `horSplitFlag` are derived as follows:

- If `partIdx` is equal to 0, the variables `nSubBlkW`, `nSubBlkH`, and `horSplitFlag` are set equal to `nBlkW`, `nBlkH`, and 0, respectively.
- Otherwise, if `partIdx` is equal to 1, the variables `nSubBlkW`, `nSubBlkH`, and `horSplitFlag` are set equal to 1, 1, and 0, respectively.
- Otherwise (`partIdx` is equal to 2), the following applies:

- The variable minSubBlkSizeFlag is derived as follows:

$$\text{minSubBlkSizeFlag} = (\text{nBlkW} \% 8 \neq 0) \mid (\text{nBlkH} \% 8 \neq 0) \quad (\text{I-302})$$

- Depending on the value of minSubBlkSizeFlag, the following applies:

- If minSubBlkSizeFlag is equal to 1, the following applies:

$$\text{horSplitFlag} = (\text{nBlkH} \% 8 \neq 0) \quad (\text{I-303})$$

- Otherwise (minSubBlkSizeFlag is equal to 0), the following applies:

$$\text{xP0} = \text{Clip3}(0, \text{pic_width_in_luma_samples} - 1, \text{xRefBlk}) \quad (\text{I-304})$$

$$\text{yP0} = \text{Clip3}(0, \text{pic_height_in_luma_samples} - 1, \text{yRefBlk}) \quad (\text{I-305})$$

$$\text{xP1} = \text{Clip3}(0, \text{pic_width_in_luma_samples} - 1, \text{xRefBlk} + \text{nBlkW} - 1) \quad (\text{I-306})$$

$$\text{yP1} = \text{Clip3}(0, \text{pic_height_in_luma_samples} - 1, \text{yRefBlk} + \text{nBlkH} - 1) \quad (\text{I-307})$$

$$\begin{aligned} \text{horSplitFlag} &= ((\text{refDepPels}[\text{xP0}][\text{yP0}] < \text{refDepPels}[\text{xP1}][\text{yP1}]) \\ &= (\text{refDepPels}[\text{xP1}][\text{yP0}] < \text{refDepPels}[\text{xP0}][\text{yP1}])) \end{aligned} \quad (\text{I-308})$$

- The variables nSubBlkW and nSubBlkH are derived as follows:

$$\text{nSubBlkW} = \text{horSplitFlag} ? 8 : 4 \quad (\text{I-309})$$

$$\text{nSubBlkH} = \text{horSplitFlag} ? 4 : 8 \quad (\text{I-310})$$

The array dSamples is derived as follows:

- For j in the range of 0 to (nBlkH / nSubBlkH – 1), inclusive, the following applies:

- For i in the range of 0 to (nBlkW / nSubBlkW – 1), inclusive, the following applies:

- The variable maxDep is set equal to –1 and modified as follows:

$$\begin{aligned} \text{xSubBlkOff} &= i * \text{nSubBlkW} \\ \text{ySubBlkOff} &= j * \text{nSubBlkH} \\ \text{xP0} &= \text{Clip3}(0, \text{pic_width_in_luma_samples} - 1, \text{xRefBlk} + \text{xSubBlkOff}) \\ \text{yP0} &= \text{Clip3}(0, \text{pic_height_in_luma_samples} - 1, \text{yRefBlk} + \text{ySubBlkOff}) \\ \text{xP1} &= \text{Clip3}(0, \text{pic_width_in_luma_samples} - 1, \text{xRefBlk} + \text{xSubBlkOff} + \text{nSubBlkW} - 1) \\ \text{yP1} &= \text{Clip3}(0, \text{pic_height_in_luma_samples} - 1, \text{yRefBlk} + \text{ySubBlkOff} + \text{nSubBlkH} - 1) \\ \text{maxDep} &= \text{Max}(\text{maxDep}, \text{refDepPels}[\text{xP0}][\text{yP0}]) \\ \text{maxDep} &= \text{Max}(\text{maxDep}, \text{refDepPels}[\text{xP0}][\text{yP1}]) \\ \text{maxDep} &= \text{Max}(\text{maxDep}, \text{refDepPels}[\text{xP1}][\text{yP0}]) \\ \text{maxDep} &= \text{Max}(\text{maxDep}, \text{refDepPels}[\text{xP1}][\text{yP1}]) \end{aligned} \quad (\text{I-311})$$

- The values of the array dSamples are derived as follows:

$$\begin{aligned} &\text{for}(\text{yOff} = 0; \text{yOff} < \text{nSubBlkH}; \text{yOff}++) \\ &\quad \text{for}(\text{xOff} = 0; \text{xOff} < \text{nSubBlkW}; \text{xOff}++) \{ \\ &\quad \quad \text{x} = \text{xSubBlkOff} + \text{xOff} \\ &\quad \quad \text{y} = \text{ySubBlkOff} + \text{yOff} \\ &\quad \quad \text{if}(\text{partIdc} == 1) \\ &\quad \quad \quad \text{dSamples}[\text{x}][\text{y}] = \text{maxDep} \\ &\quad \quad \text{else} \\ &\quad \quad \quad \text{dSamples}[\text{x}][\text{y}] = \text{DepthToDisparityB}[\text{refViewIdx}][\text{maxDep}] \\ &\quad \quad \} \end{aligned} \quad (\text{I-312})$$

I.8.6 Scaling, transformation and array construction process prior to deblocking filter process

The specifications in clause 8.6 apply.

I.8.7 In-loop filter process

The specifications in clause 8.7 apply.

I.9 Parsing process

I.9.1 General

The specifications in clause 9.1 apply with the following modifications.

- All references to the process specified in clause 9.3 are replaced with references to the process specified in clause I.9.3.

I.9.2 Parsing process for 0-th order Exp-Golomb codes

The specifications in clause 9.2 and all its subclauses apply.

I.9.3 CABAC parsing process for slice segment data

I.9.3.1 General

The specifications in clause 9.3.1 apply with the following modifications.

- All references to the process specified in clauses 7.3.8.1 to 7.3.8.12 are replaced with references to the process specified in clauses I.7.3.8.1 to I.7.3.8.12, respectively.
- All invocations of the process specified in clause 9.3.2 to 9.3.4 are replaced with invocations of the process specified in clause I.9.3.2 to I.9.3.4.
- All invocations of the process specified in clause 9.3.2.4 are replaced with invocations of the process specified in clause I.9.3.2.3.
- All invocations of the process specified in clause 9.3.2.6 are replaced with invocations of the process specified in clause I.9.3.2.5.

I.9.3.2 Initialization process

I.9.3.2.1 General

The specifications in clause 9.3.2.1 apply with the following modifications.

- All invocations of the process specified in clause 9.3.2.2 are replaced with invocations of the process specified in clause I.9.3.2.2.
- All invocations of the process specified in clause 9.3.2.5 are replaced with invocations of the process specified in clause I.9.3.2.4.
- All invocations of the process specified in clause 9.3.2.6 are replaced with invocations of the process specified in clause I.9.3.2.5.

I.9.3.2.2 Initialization process for context variables

The specifications in clause 9.3.2.2 apply with the following modifications.

- All references to the process specified in clauses 7.3.8.1 to 7.3.8.12 are replaced with references to the process specified in clauses I.7.3.8.1 to I.7.3.8.12.
- Table I.4 is appended to the end of Table 9-4.
- Table I.5 to Table I.14 are appended to the end of the clause.

Table I.4 – Association of ctxIdx and syntax elements for each initializationType in the initialization process

Syntax structure	Syntax element	ctxTable	initType		
			0	1	2
coding_unit()	skip_intra_flag	Table I.5	0	1	2
intra_mode_ext()	no_dim_flag	Table I.6	0	1	2
	depth_intra_mode_idx_flag	Table I.7	0	1	2
cu_extension()	skip_intra_mode_idx	Table I.8	0	1	2
	dbbp_flag	Table I.9	0	1	2
	dc_only_flag	Table I.10	0	1	2
	iv_res_pred_weight_idx	Table I.11		0..2	3..5
	illu_comp_flag	Table I.12		0	1
depth_dcs()	depth_dc_present_flag	Table I.13	0	1	2
	depth_dc_abs	Table I.14	0	1	2

Table I.5 – Values of initValue for skip_intra_flag ctxIdx

Initialization variable	ctxIdx of skip_intra_flag ctxIdx		
	0	1	2
initValue	185	185	185

Table I.6 – Values of initValue for no_dim_flag ctxIdx

Initialization variable	ctxIdx of no_dim_flag ctxIdx		
	0	1	2
initValue	154	141	155

Table I.7 – Values of initValue for depth_intra_mode_idx_flag ctxIdx

Initialization variable	ctxIdx of depth_intra_mode_idx_flag		
	0	1	2
initValue	154	154	154

Table I.8 – Values of initValue for skip_intra_mode_idx ctxIdx

Initialization variable	ctxIdx of skip_intra_mode_idx ctxIdx		
	0	1	2
initValue	137	137	137

Table I.9 – Values of initValue for dbbp_flag ctxIdx

Initialization variable	ctxIdx of dbbp_flag		
	0	1	2
initValue	154	154	154

Table I.10 – Values of initValue for dc_only_flag ctxIdx

Initialization variable	ctxIdx of dc_only_flag ctxIdx		
	0	1	2
initValue	154	154	154

Table I.11 – Values of initValue for iv_res_pred_weight_idx ctxIdx

Initialization variable	ctxIdx of iv_res_pred_weight_idx					
	0	1	2	3	4	5
initValue	162	153	162	162	153	162

Table I.12 – Values of initValue for illu_comp_flag ctxIdx

Initialization variable	ctxIdx of illu_comp_flag	
	0	1
initValue	154	154

Table I.13 – Values of initValue for depth_dc_present_flag ctxIdx

Initialization variable	ctxIdx of depth_dc_present_flag		
	0	1	2
initValue	0	0	64

Table I.14 – Values of initValue for depth_dc_abs ctxIdx

Initialization variable	ctxIdx of depth_dc_abs		
	0	1	2
initValue	154	154	154

I.9.3.2.3 Storage process for context variables and Rice parameter initialization states

The specifications in clause 9.3.2.4 apply with the following modifications

- All references to the process specified in clauses 7.3.8.1 to 7.3.8.12 are replaced with references to the process specified in clauses I.7.3.8.1 to I.7.3.8.12.

I.9.3.2.4 Synchronization process for context variables and Rice parameter initialization states

The specifications in clause 9.3.2.5 apply with the following modifications

- All references to the process specified in clauses 7.3.8.1 to 7.3.8.12 are replaced with references to the process specified in clauses I.7.3.8.1 to I.7.3.8.12.

I.9.3.2.5 Initialization process for the arithmetic decoding engine

The specifications in clause 9.3.2.6 apply.

I.9.3.3 Binarization process**I.9.3.3.1 General**

The specifications in clause 9.3.3.1 apply with the following modifications.

- All references to processes clauses 9.3.3.2 to 9.3.3.11 are replaced with references to the processes specified in clauses I.9.3.3.2 and I.9.3.3.10 respectively.
- Table I.15 is appended to the end of Table 9-43.

Table I.15 – Syntax elements and associated binarizations

Syntax structure	Syntax element	Binarization	
		Process	Input parameters
coding_unit()	skip_intra_flag	FL	cMax = 1
intra_mode_ext()	no_dim_flag	FL	cMax = 1
	depth_intra_mode_idx_flag	FL	cMax = 1
	wedge_full_tab_idx	FL	cMax = NumWedgePattern[log2PbSize] – 1
cu_extension()	skip_intra_mode_idx	TR	cMax = 3, cRiceParam = 0
	dbbp_flag	FL	cMax = 1
	dc_only_flag	FL	cMax = 1
	iv_res_pred_weight_idx	TR	cMax = 2, cRiceParam = 0
	illu_comp_flag	FL	cMax = 1
depth_dcs()	depth_dc_present_flag	FL	cMax = 1
	depth_dc_abs	I.9.3.3.11	-
	depth_dc_sign_flag	FL	cMax = 1

I.9.3.3.2 Truncated Rice (TR) binarization process

The specifications in clause 9.3.3.2 apply.

I.9.3.3.3 k-th order Exp-Golomb (EGk) binarization process

The specifications in clause 9.3.3.3 apply.

I.9.3.3.4 Limited k-th order Exp-Golomb (EGk) binarization process

The specifications in clause 9.3.3.4 apply.

I.9.3.3.5 Fixed-length (FL) binarization process

The specifications in clause 9.3.3.5 apply.

I.9.3.3.6 Binarization process for part_mode

Inputs to this process are a request for a binarization for the syntax element part_mode, a luma location (xCb, yCb), specifying the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture, a variable log2CbSize specifying the current luma coding block size, and the variable partPredIdc indicating a reference partition mode.

Output of this process is the binarization of the syntax element.

The binarization for the syntax element part_mode is specified in Table I.16 depending on the values of CuPredMode[xCb][yCb], log2CbSize, and partPredIdc.

Table I.16 – Binarization for part_mode

	part_mode	partPredIdc	PartMode	Bin string
--	-----------	-------------	----------	------------

CuPredMode [xCb][yCb]				log2CbSize == MinCbLog2SizeY			
				0		1	
				amp_enabled_flag		log2CbSize > 3	
				0	1	0	1
MODE_INTRA	0	0	PART_2Nx2N	-	-	1	1
	1	0	PART_NxN	-	-	0	0
MODE_INTER	0	0, 1, 2	PART_2Nx2N	1	1	1	1
	1	0	PART_2NxN	01	011	01	01
	2	0	PART_Nx2N	00	001	00	001
	3	0	PART_NxN	-	-	-	000
	4	0	PART_2NxN	-	0100	-	-
	5	0	PART_2NxN	-	0101	-	-
	6	0	PART_nLx2N	-	0000	-	-
	7	0	PART_nRx2N	-	0001	-	-
	1	1	PART_2NxN	0	01	0	0
	2	1	PART_2NxN	-	000	-	-
	3	1	PART_2NxN	-	001	-	-
	1	2	PART_Nx2N	0	01	0	0
	2	2	PART_nLx2N	-	000	-	-
	3	2	PART_nRx2N	-	001	-	-

I.9.3.3.7 Binarization process for intra_chroma_pred_mode

The specifications in clause 9.3.3.8 apply.

I.9.3.3.8 Binarization process for inter_pred_idc

The specifications in clause 9.3.3.9 apply.

I.9.3.3.9 Binarization process for cu_qp_delta_abs

The specifications in clause 9.3.3.10 apply.

I.9.3.3.10 Binarization process for coeff_abs_level_remaining[]

The specifications in clause 9.3.3.11 apply.

I.9.3.3.11 Binarization process for depth_dc_abs

Input to this process is a request for a binarization for the syntax element depth_dc_abs.

Output of this process is the binarization of the syntax element.

The binarization of the syntax element depth_dc_abs is a concatenation of a prefix bin string and (when present) a suffix bin string.

For the derivation of the prefix bin string, the following applies:

- The prefix value of depth_dc_abs, prefixVal, is derived as follows:

$$\text{prefixVal} = \text{Min}(\text{depth_dc_abs}, 3) \quad (\text{I-313})$$

- The prefix bin string is specified by invoking the TR binarization process as specified in clause 9.3.3.2 for prefixVal with cMax = 3 and cRiceParam = 0.

When prefixVal is greater than 2, the suffix bin string is present and it is derived as follows:

- The suffix value of depth_dc_abs, suffixVal, is derived as follows:

$$\text{suffixVal} = \text{depth_dc_abs} - 3 \quad (\text{I-314})$$

- The suffix bin string is specified by invoking the k-th order EGk binarization process as specified in clause 9.3.3.3 for suffixVal with the Exp-Golomb order k set equal to 0.

I.9.3.4 Decoding process flow

I.9.3.4.1 General

The specifications in clause 9.3.4.1 apply with the following modifications.

- All references to the process specified in clause 9.3.3 are replaced with references to the process specified in clause I.9.3.3.
- All invocations of the process specified in clause 9.3.4.2 are replaced with invocations of the process specified in clause I.9.3.4.2.
- All invocations of the process specified in clause 9.3.4.3 are replaced with invocations of the process specified in clause I.9.3.4.3.

I.9.3.4.2 Derivation process for ctxTable, ctxIdx and bypassFlag

I.9.3.4.2.1 General

The specifications in clause 9.3.4.2.1 apply with the following modifications:

- All references to the process specified in clause 9.3.4.2.2 are replaced by references to the process specified in clause I.9.3.4.2.2.
- Table I.17 is appended to the end of Table 9-48.

Table I.17 – Assignment of ctxInc to syntax elements with context coded bins

Syntax element	binIdx					
	0	1	2	3	4	>= 5
skip_intra_flag	0	na	na	na	na	na
no_dim_flag	0	na	na	na	na	na
depth_intra_mode_idx_flag	0	na	na	na	na	na
wedge_full_tab_idx	bypass	bypass	bypass	bypass	bypass	bypass
skip_intra_mode_idx	0	bypass	bypass	na	na	na
dbbp_flag	0	na	na	na	na	na
dc_only_flag	0	na	na	na	na	na
iv_res_pred_weight_idx	0, 1 (clause I.9.3.4.2.2)	2	na	na	na	na
illu_comp_flag	0	na	na	na	na	na
depth_dc_present_flag	0	na	na	na	na	na
depth_dc_abs	0	0	0	bypass	bypass	bypass
depth_dc_sign_flag	bypass	0	0	0	0	0

I.9.3.4.2.2 Derivation process of ctxInc using left and above syntax elements

The specifications in clause 9.3.4.2.2 apply with the following modifications and additions:

- Table I.18 is appended to the end of Table 9-49.

Table I.18 – Specification of ctxInc using left and above syntax elements

Syntax element	condL	condA	ctxInc
iv_res_pred_weight_idx[x0][x0]	iv_res_pred_weight_idx[xNbL][yNbL]		(condL && availableL)

- The assignment of ctxInc for the syntax elements iv_res_pred_weight_idx[x0][y0] is specified in Table 9-49.

I.9.3.4.2.3 Derivation process of ctxInc for the syntax elements last_sig_coeff_x_prefix and last_sig_coeff_y_prefix

The specifications in clause 9.3.4.2.3 apply.

I.9.3.4.2.4 Derivation process of ctxInc for the syntax element coded_sub_block_flag

The specifications in clause 9.3.4.2.4 apply.

I.9.3.4.2.5 Derivation process of ctxInc for the syntax element sig_coeff_flag

The specifications in clause 9.3.4.2.5 apply.

I.9.3.4.2.6 Derivation process of ctxInc for the syntax element coeff_abs_level_greater1_flag

The specifications in clause 9.3.4.2.6 apply.

I.9.3.4.2.7 Derivation process of ctxInc for the syntax element coeff_abs_level_greater2_flag

The specifications in clause 9.3.4.2.7 apply.

I.9.3.4.3 Arithmetic decoding process

The specifications in clause 9.3.4.3 and all its subclauses apply with the following modifications:

- All references to the process specified in clause 9.3.4.2 are replaced with references to the process specified in clause I.9.3.4.2.

I.9.3.5 Arithmetic encoding process (informative)

The specifications in clause 9.3.5 and all its subclauses apply with the following modifications:

- All references to the process specified in clause 9.3.4.3 are replaced with references to the process specified in clause I.9.3.4.3.

I.10 Specification of bitstream subsets

The specifications in clause G.10 apply.

I.11 Profiles, tiers, and levels

I.11.1 Profiles

I.11.1.1 3D Main profile

For a layer in an output operation point associated with an OLS in a bitstream, the layer being conforming to the 3D Main profile, the following applies:

- Let olsIdx be the OLS index of the OLS, the sub-bitstream subBitstream and the base layer sub-bitstream baseBitstream are derived as specified in clause F.11.3.

When vps_base_layer_internal_flag is equal to 1, the base layer sub-bitstream baseBitstream shall obey the following constraints:

- The base layer sub-bitstream baseBitstream shall be indicated to conform to the Main profile.

The sub-bitstream subBitstream shall obey the following constraints:

- All active VPSs shall have vps_num_rep_formats_minus1 in the range of 0 to 15, inclusive.
- All active SPSs for a layer with nuh_layer_id equal to i and DepthLayerFlag[i] equal to 0 in subBitstream shall have chroma_format_idc equal to 1 only.
- All active SPSs for a layer with nuh_layer_id equal to i and DepthLayerFlag[i] equal to 1 in subBitstream shall have chroma_format_idc equal to 0 only.
- All active SPSs for layers in subBitstream shall have transform_skip_rotation_enabled_flag, transform_skip_context_enabled_flag, implicit_rdpem_enabled_flag, explicit_rdpem_enabled_flag, extended_precision_processing_flag, intra_smoothing_disabled_flag, high_precision_offsets_enabled_flag, persistent_rice_adaptation_enabled_flag, and cabac_bypass_alignment_enabled_flag, when present, equal to 0 only.
- CtbLog2SizeY derived from all active SPSs for layers in subBitstream shall be in the range of 4 to 6, inclusive.
- All active PPSs for layers in subBitstream shall have log2_max_transform_skip_block_size_minus2 and chroma_qp_offset_list_enabled_flag, when present, equal to 0 only.
- ScalabilityId[j][smIdx] derived according to any active VPS shall be equal to 0 for any smIdx value not equal to 0 or 1 and for any value of j such that layer_id_in_nuh[j] is among layerIdListTarget that was used to derive subBitstream.

- All active VPSs shall have `alt_output_layer_flag[olsIdx]` equal to 0 only.
- When `ViewOrderIdx[i]` or `DepthLayerFlag[i]` derived according to any active VPS is greater than 0 for the layer with `nuh_layer_id` equal to `i` in `subBitstream`, `num_ref_loc_offsets` shall be equal to 0 in each active PPS for that layer.
- When `ViewOrderIdx[i]` or `DepthLayerFlag[i]` derived according to any active VPS is greater than 0 for the layer with `nuh_layer_id` equal to `i` in `subBitstream`, the values of `pic_width_in_luma_samples` and `pic_height_in_luma_samples` in each active SPS for that layer shall be equal to the values of `pic_width_in_luma_samples` and `pic_height_in_luma_samples`, respectively, in each active SPS for all reference layers of that layer.
- For a layer with `nuh_layer_id` `iNuhLId` equal to any value included in `layerIdListTarget` that was used to derive `subBitstream`, the value of `NumRefLayers[iNuhLId]`, which specifies the total number of direct and indirect reference layers and is derived as specified in F.7.4.3.1, shall be less than or equal to 9.
- All active SPSs for layers in `subBitstream` shall have `sps_range_extension_flag` and `sps_scc_extension_flag` equal to 0 only.
- All active PPSs for layers in `subBitstream` shall have `pps_range_extension_flag` and `pps_scc_extension_flag` equal to 0 only.
- All active SPSs for layers in `subBitstream` shall have `bit_depth_luma_minus8` equal to 0 only.
- All active SPSs for layers in `subBitstream` shall have `bit_depth_chroma_minus8` equal to 0 only.
- All active PPSs for layers in `subBitstream` shall have `colour_mapping_enabled_flag` equal to 0 only.
- When an active PPS for any layer in `subBitstream` has `tiles_enabled_flag` equal to 1, it shall have `entropy_coding_sync_enabled_flag` equal to 0.
- When an active PPS for any layer in `subBitstream` has `tiles_enabled_flag` equal to 1, `ColumnWidthInLumaSamples[i]` shall be greater than or equal to 256 for all values of `i` in the range of 0 to `num_tile_columns_minus1`, inclusive, and `RowHeightInLumaSamples[j]` shall be greater than or equal to 64 for all values of `j` in the range of 0 to `num_tile_rows_minus1`, inclusive.
- The number of times `read_bits(1)` is called in clauses 9.3.4.3.3 and 9.3.4.3.4 when parsing `coding_tree_unit()` data for any CTU shall be less than or equal to $5 * \text{RawCtuBits} / 3$.
- `general_level_idc` and `sub_layer_level_idc[i]` for all values of `i` in active SPSs for any layer in `subBitstream` shall not be equal to 255 (which indicates level 8.5).
- The tier and level constraints specified for the 3D Main profile in clause I.11.2 shall be fulfilled.
- For any active VPS, `ViewOrderIdx[i]` shall be greater than `ViewOrderIdx[j]` for any values of `i` and `j` among `layerIdListTarget` that was used to derive `subBitstream` such that `DepthLayerFlag[i]` is equal to `DepthLayerFlag[j]` and `i` is greater than `j`.
- For any active VPS, `LayerIdxInVps[i]` shall be equal to `LayerIdxInVps[j] + 1` for any values of `i` and `j` among `layerIdListTarget` that was used to derive `subBitstream` such that `ViewOrderIdx[i]` is equal to `ViewOrderIdx[j]`, `DepthLayerFlag[i]` is equal to 1, and `DepthLayerFlag[j]` is equal to 0.

In the remainder of this clause and clause I.11.2, all syntax elements in the `profile_tier_level()` syntax structure refer to those in the `profile_tier_level()` syntax structure associated with the layer.

Conformance of a layer in an output operation point associated with an OLS in a bitstream to the 3D Main profile is indicated as follows:

- If `OpTid` of the output operation point is equal to `vps_max_sub_layer_minus1`, the conformance is indicated by `general_profile_idc` being equal to 8 or `general_profile_compatibility_flag[8]` being equal to 1, and `general_max_12bit_constraint_flag` being equal to 1, `general_max_10bit_constraint_flag` being equal to 1, `general_max_8bit_constraint_flag` being equal to 1, `general_max_422chroma_constraint_flag` being equal to 1, `general_max_420chroma_constraint_flag` being equal to 1, `general_max_monochrome_constraint_flag` being equal to 0, `general_intra_constraint_flag` being equal to 0, and `general_one_picture_only_constraint_flag` being equal to 0, and `general_lower_bit_rate_constraint_flag` being equal to 1.
- Otherwise (`OpTid` of the output operation point is less than `vps_max_sub_layer_minus1`), the conformance is indicated by `sub_layer_profile_idc[OpTid]` being equal to 8 or `sub_layer_profile_compatibility_flag[OpTid][8]` being equal to 1, and `sub_layer_max_12bit_constraint_flag[OpTid]` being equal to 1, `sub_layer_max_10bit_constraint_flag[OpTid]` being equal to 1, `sub_layer_max_8bit_constraint_flag[OpTid]` being equal to 1, `sub_layer_max_422chroma_constraint_flag[OpTid]` being equal to 1, `sub_layer_max_420chroma_constraint_flag[OpTid]` being equal to 1, `sub_layer_max_monochrome_constraint_flag[OpTid]` being equal to 0, `sub_layer_intra_constraint_flag[OpTid]` being equal to 0, and `sub_layer_one_picture_only_constraint_flag[OpTid]` being equal to 0, and `sub_layer_lower_bit_rate_constraint_flag[OpTid]` being equal to 1.

sub_layer_max_monochrome_constraint_flag[OpTid] being equal to 0, sub_layer_intra_constraint_flag[OpTid] being equal to 0, and sub_layer_one_picture_only_constraint_flag[OpTid] being equal to 0, and sub_layer_lower_bit_rate_constraint_flag[OpTid] being equal to 1.

I.11.2 Tiers and levels

The specification in sub-clause G.11.2 and its sub-clauses apply with the following modifications:

- "Multiview Main profile" is replaced by "3D Main profile"

I.11.3 Decoder capabilities

When a decoder conforms to any profile specified in Annex I, it shall also have the INBLD capability specified in clause F.11.1.

Clause F.11.2 specifies requirements for a decoder conforming to any profile specified in Annex I.

I.12 Byte stream format

The specifications in clause G.12 apply.

I.13 Hypothetical reference decoder

The specifications in clause G.13 apply.

I.14 Supplemental enhancement information

I.14.1 General

The specifications in clause G.14.1 apply.

I.14.2 SEI payload syntax

I.14.2.1 General SEI payload syntax

The specifications in clause G.14.2.1 apply.

I.14.2.2 Annex D, Annex F, and Annex G SEI message syntax for 3D high efficiency video coding

The specifications in clauses G.14.2.2 through G.14.2.7 apply.

I.14.2.3 Alternative depth information SEI message syntax

alternative_depth_info(payloadSize) {	Descriptor
alternative_depth_info_cancel_flag	u(1)
if(alternative_depth_info_cancel_flag == 0) {	
depth_type	u(2)
if(depth_type == 0) {	
num_constituent_views_gvd_minus1	ue(v)
depth_present_gvd_flag	u(1)
z_gvd_flag	u(1)
intrinsic_param_gvd_flag	u(1)
rotation_gvd_flag	u(1)
translation_gvd_flag	u(1)
if(z_gvd_flag)	
for(i = 0; i <= num_constituent_views_gvd_minus1 + 1; i++) {	
sign_gvd_z_near_flag[i]	u(1)
exp_gvd_z_near[i]	u(7)
man_len_gvd_z_near_minus1[i]	u(5)
man_gvd_z_near[i]	u(v)
sign_gvd_z_far_flag[i]	u(1)
exp_gvd_z_far[i]	u(7)

man_len_gvd_z_far_minus1[i]	u(5)
man_gvd_z_far[i]	u(v)
}	
if(intrinsic_param_gvd_flag) {	
prec_gvd_focal_length	ue(v)
prec_gvd_principal_point	ue(v)
}	
if(rotation_gvd_flag)	
prec_gvd_rotation_param	ue(v)
if(translation_gvd_flag)	
prec_gvd_translation_param	ue(v)
for(i = 0; i <= num_constituent_views_gvd_minus1 + 1; i++) {	
if(intrinsic_param_gvd_flag) {	
sign_gvd_focal_length_x[i]	u(1)
exp_gvd_focal_length_x[i]	u(6)
man_gvd_focal_length_x[i]	u(v)
sign_gvd_focal_length_y[i]	u(1)
exp_gvd_focal_length_y[i]	u(6)
man_gvd_focal_length_y[i]	u(v)
sign_gvd_principal_point_x[i]	u(1)
exp_gvd_principal_point_x[i]	u(6)
man_gvd_principal_point_x[i]	u(v)
sign_gvd_principal_point_y[i]	u(1)
exp_gvd_principal_point_y[i]	u(6)
man_gvd_principal_point_y[i]	u(v)
}	
if(rotation_gvd_flag)	
for(j = 0; j < 3; j++) /* row */	
for(k = 0; k < 3; k++) { /* column */	
sign_gvd_r[i][j][k]	u(1)
exp_gvd_r[i][j][k]	u(6)
man_gvd_r[i][j][k]	u(v)
}	
if(translation_gvd_flag) {	
sign_gvd_t_x[i]	u(1)
exp_gvd_t_x[i]	u(6)
man_gvd_t_x[i]	u(v)
}	
}	
}	
if(depth_type == 1) {	
min_offset_x_int	se(v)
min_offset_x_frac	u(8)
max_offset_x_int	se(v)
max_offset_x_frac	u(8)
offset_y_present_flag	u(1)
if(offset_y_present_flag){	
min_offset_y_int	se(v)

min_offset_y_frac	u(8)
max_offset_y_int	se(v)
max_offset_y_frac	u(8)
}	
warp_map_size_present_flag	u(1)
if(warp_map_size_present_flag) {	
warp_map_width_minus2	ue(v)
warp_map_height_minus2	ue(v)
}	
}	
}	
}	

I.14.3 SEI payload semantics

I.14.3.1 General SEI payload semantics

The specifications in clause G.14.3.1 apply with the following modifications and additions:

The list VclAssociatedSeiList is set to consist of the payloadType values 2, 3, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 131, 132, 134 to 152, inclusive, 154 to 159, inclusive, 161, 165, 167, 168, 177, 178, 179, 200 to 202, inclusive, and 205.

The list PicUnitRepConSeiList is set to consist of the payloadType values 0, 1, 2, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 129, 131, 132, 133, 135 to 152, inclusive, 154 to 168, inclusive, 176 to 181, inclusive, 200 to 202, inclusive, and 205.

The semantics and persistence scope for each SEI message specified in this annex are specified in the semantics specification for each particular SEI message in the subclauses of this clause.

NOTE – Persistence information for SEI messages specified in this annex is informatively summarized in Table I.19.

Table I.19 – Persistence scope of SEI messages (informative)

SEI message	Persistence scope
Alternative depth information	Specified by the semantics of the SEI message

I.14.3.2 Annex D, Annex F, and Annex G SEI message semantics for 3D high efficiency video coding

I.14.3.2.1 General

The specifications of clause G.14.3.2 and its subclauses apply with the modifications specified in clause I.14.3.2.2.

I.14.3.2.2 Scalable nesting SEI message semantics for 3D high efficiency video coding

The specifications of clause G.14.3.2.2 apply with the following additions:

An SEI message that has payloadType equal to 181 (Alternative depth information) shall not be directly contained in a scalable nesting SEI message.

I.14.3.3 Alternative depth information SEI message semantics

The alternative depth information SEI message indicates that the decoded depth samples are interpreted as being of an alternative depth format. To discriminate different alternative depth formats, the depth_type syntax element is used.

Let currPic be a picture in the current access unit containing the alpha channel information SEI message. The information of the alternative depth information SEI message persists in output order until any of the following are true:

- A new CVS begins.
- The bitstream ends.
- A picture picB in an access unit containing an alternative depth information SEI message is output having PicOrderCnt(picB) greater than PicOrderCnt(currPic), where PicOrderCnt(picB) and PicOrderCnt(currPic) are the PicOrderCntVal values of picB and currPic, respectively, immediately after the invocation of the decoding process

for picture order count for picB.

alternative_depth_info_cancel_flag equal to 1 indicates that the SEI message cancels the persistence of any previous alternative depth information SEI message in output order. **alternative_depth_info_cancel_flag** equal to 0 indicates that alternative depth information follows.

depth_type identifies an alternative depth type according to Table I.20. **depth_type** equal to 0 indicates that global view and depth (GVD) information is present in this SEI message. **depth_type** equal to 1 indicates that the decoded depth samples can be used to derive a warp map and view synthesis can be performed by image-domain warping. Values of **depth_type** that are not listed in Table I.20 are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore alternative depth information SEI messages that contain reserved values of **depth_type**.

Table I.20 – Interpretation of depth_type

Value	Description
0	Global view and depth (GVD)
1	Warp map

When the SEI message signals GVD information, the pictures of a view with ViewIdx greater than 0 contain samples of multiple subsampled pictures (subsequently denoted as constituent pictures), which are located in a spatial packing arrangement. This GVD information can be used after the decoder output to appropriately rearrange the samples in order to produce additional views that are appropriate for display or other purposes (which are outside the scope of this Recommendation | International Standard).

Layers that are referred to by this GVD information shall conform to the Main Profile, the Multiview Main profile, or the 3D Main profile, as specified in Annexes A, G, and I, respectively. The depth representation type is defined in the depth representation information SEI message that shall be always present together with the GVD information.

When the CVS contains an alternative depth information SEI message with **depth_type** equal to 0, and the first access unit in the CVS is an IRAP access unit, there shall be an alternative depth information SEI message with **depth_type** equal to 0 in the first access unit of the CVS. When GVD information is present, the following applies:

- The CVS shall represent two views.
- A texture layer and a depth layer shall both be present for the view with ViewIdx equal to 0.
- A texture layer shall be present and a depth layer may be present for the view with ViewIdx greater than 0.
- The value of NumDirectRefLayers[LayerIdxInVps[depLayerId]] shall be equal to 0 for any value of depLayerId among nuh_layer_id values of layers of the view with ViewIdx equal to 0 or the view with ViewIdx greater than 0.
- The pictures of the view with ViewIdx equal to 0 shall represent full resolution pictures.
- Each picture of the view with ViewIdx greater than 0 shall contain a packing arrangement of 1 to 4 sub-sampled pictures of constituent views. Such sub-sampled pictures are denoted as constituent pictures.
- All constituent pictures shall have a width and a height equal to $(\text{pic_width_in_luma_samples} / 2)$ and $(\text{pic_height_in_luma_samples} / 2)$ in luma samples, respectively.

The variable *i* (with a value in the range of 1 to num_constituent_views_minus1 + 1, inclusive) in the alternative_depth_info() syntax structure is an index that is associated with the location of the constituent picture in the picture of the view with ViewIdx greater than 0, as specified in Table I.21. With regard to the remaining syntax elements in the alternative_depth_info() syntax structure, *i* equal to 0 refers to the parameters of the view with ViewIdx equal to 0 and *i* greater than 0 refers to the parameter of one of the constituent views.

NOTE 1 – The ViewIdx of the view with ViewIdx greater than 0 is not used for identification of constituent views in the context of this SEI message, but the variable *i*.

Table I.21 – Locations of the top-left luma samples of constituent pictures packed in a picture with ViewIdx greater than 0 relative to the top-left luma sample of this picture

Constituent picture index <i>i</i>	Location of the top-left luma sample in a picture with ViewIdx greater than 0
1	(0, 0)
2	(0, pic_height_in_luma_samples / 2)

3	(pic_width_in_luma_samples / 2, 0)
4	(pic_width_in_luma_samples / 2, pic_height_in_luma_samples / 2)

num_constituent_views_gvd_minus1 plus 1 specifies the number of constituent pictures packed into each picture of the view with ViewIdx greater than 0. num_constituent_views_gvd_minus1 shall be in the range of 0 to 3, inclusive.

depth_present_gvd_flag equal to 1 specifies that the depth layer for the view with ViewIdx greater than 0 is present and contains constituent pictures with a packing arrangement as described above. depth_present_gvd_flag equal to 0 specifies that the depth layer for the view with ViewIdx greater than 0 is not present.

Each constituent picture of a depth layer for the view with ViewIdx greater than 0 is associated with a constituent picture of the texture layer for the view with ViewIdx greater than 0 in the same relative location.

NOTE 2 – The following SEI message parameters can be used along with the decoded pictures of the depth layers to project samples from the view with ViewIdx equal to 0 into the co-ordinates of constituent views such that the reconstructed views can be generated by combining the projected samples and the samples from the constituent views.

The function binToFp(s, e, n, v) is specified as follows:

$$\text{binToFp}(s, e, n, v) = (-1)^s * (e == 0 ? (2^{-(30+v)} * n) : (2^{e-31} * (1 + n \div 2^v))) \quad (\text{I-315})$$

NOTE 3 – The above specification is similar to what is found in IEC 60559:1989.

z_gvd_flag equal to 1 indicates the presence of the syntax elements sign_gvd_z_near_flag[i], exp_gvd_z_near[i], man_len_gvd_z_near_minus1[i], man_gvd_z_near[i], sign_gvd_z_far_flag[i], exp_gvd_z_far[i], man_len_gvd_z_far_minus1[i], and man_gvd_z_far[i], for i in the range of 0 to num_constituent_views_minus1 + 1, inclusive. z_gvd_flag equal to 0 indicates that these syntax elements are not present.

intrinsic_param_gvd_flag equal to 1 indicates the presence of intrinsic camera parameter syntax elements. intrinsic_param_gvd_flag equal to 0 indicates that these syntax elements are not present.

rotation_gvd_flag equal to 1 indicates the presence of rotation camera parameter syntax elements. rotation_gvd_flag equal to 0 indicates that these syntax elements are not present. When rotation_gvd_flag is equal to 0, a default rotation camera parameter of a unit matrix value is inferred.

translation_gvd_flag equal to 1 indicates the presence of horizontal translation camera parameter syntax elements. translation_gvd_flag equal to 0 indicates that these syntax elements are not present.

sign_gvd_z_near_flag[i] equal to 0 indicates that the sign of the nearest depth value of the i-th camera is positive. sign_gvd_z_near_flag[i] equal to 1 indicates that the sign of the nearest depth value of the i-th camera is negative.

exp_gvd_z_near[i] specifies the exponent part of the nearest depth value of the i-th camera. The value of exp_gvd_z_near[i] shall be in the range of 0 to 126, inclusive. The value 127 is reserved for future use by ITU-T | ISO/IEC. When exp_gvd_z_near[i] is equal to 127, the value of zNear[i] is unspecified.

man_len_gvd_z_near_minus1[i] plus 1 specifies the length in bits of the mantissa of the nearest depth value of the i-th camera. The value of man_len_gvd_z_near_minus1[i] shall be in the range of 0 to 31, inclusive.

man_gvd_z_near[i] specifies the mantissa part of the nearest depth value of the i-th camera. The length of man_gvd_z_near[i] syntax elements is man_len_gvd_z_near_minus1[i] + 1 bits.

When exp_gvd_z_near[i] is not equal to 127, zNear[i] is set equal to binToFp(sign_gvd_z_near_flag[i], exp_gvd_z_near[i], man_gvd_z_near[i], man_len_gvd_z_near_minus1[i] + 1).

sign_gvd_z_far_flag[i] equal to 0 indicates that the sign of the farthest depth value of the i-th camera is positive. sign_gvd_z_far_flag[i] equal to 1 indicates that the sign of the farthest depth value of the i-th camera is negative.

exp_gvd_z_far[i] specifies the exponent part of the farthest depth value of the i-th camera. The value of exp_gvd_z_far[i] shall be in the range of 0 to 126, inclusive. The value 127 is reserved for future use by ITU-T | ISO/IEC. When exp_gvd_z_far[i] is equal to 127, the value of zFar[i] is unspecified.

man_len_gvd_z_far_minus1[i] plus 1 specifies the length in bits of the mantissa of the farthest depth value of the i-th camera. The value of man_len_gvd_z_far_minus1[i] shall be in the range of 0 to 31, inclusive.

man_gvd_z_far[i] specifies the mantissa part of the farthest depth value of the i-th camera. The length of man_gvd_z_far[i] syntax elements is man_len_gvd_z_far_minus1[i] + 1 bits.

When exp_gvd_z_far[i] is not equal to 127, zFar[i] is set equal to binToFp(sign_gvd_z_far_flag[i], exp_gvd_z_far[i], man_gvd_z_far[i], man_len_gvd_z_far_minus1[i] + 1).

prec_gvd_focal_length specifies the exponent of the maximum allowable truncation error for focalLengthX[i] and

focalLengthY[i] as given by $2^{-\text{prec_gvd_focal_length}}$. The value of prec_gvd_focal_length shall be in the range of 0 to 31, inclusive.

prec_gvd_principal_point specifies the exponent of the maximum allowable truncation error for principalPointX[i] and principalPointY[i] as given by $2^{-\text{prec_gvd_principal_point}}$. The value of prec_gvd_principal_point shall be in the range of 0 to 31, inclusive.

prec_gvd_rotation_param specifies the exponent of the maximum allowable truncation error for $r[i][j][k]$ as given by $2^{-\text{prec_gvd_rotation_param}}$. The value of prec_gvd_rotation_param shall be in the range of 0 to 31, inclusive.

prec_gvd_translation_param specifies the exponent of the maximum allowable truncation error for $tX[i]$ as given by $2^{-\text{prec_gvd_translation_param}}$. The value of prec_gvd_translation_param shall be in the range of 0 to 31, inclusive.

sign_gvd_focal_length_x[i] equal to 0 indicates that the sign of the focal length of the i-th camera in the horizontal direction is positive. sign_gvd_focal_length_x[i] equal to 1 indicates that the sign of the focal length of the i-th camera in the horizontal direction is negative.

exp_gvd_focal_length_x[i] specifies the exponent part of the focal length of the i-th camera in the horizontal direction. The value of exp_gvd_focal_length_x[i] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. When exp_gvd_focal_length_x[i] is equal to 63, the value of focal length of the horizontal direction for the i-th camera is unspecified.

man_gvd_focal_length_x[i] specifies the mantissa part of the focal length of the i-th camera in the horizontal direction. The length v of the man_gvd_focal_length_x[i] syntax element is determined as follows:

- If exp_gvd_focal_length_x[i] is equal to 0, the length v is set equal to $\text{Max}(0, \text{prec_gvd_focal_length} - 30)$.
- Otherwise (exp_gvd_focal_length_x[i] is in the range of 1 to 62, inclusive), the length v is $\text{Max}(0, \text{exp_gvd_focal_length_x}[i] + \text{prec_gvd_focal_length} - 31)$.

When exp_gvd_focal_length_x[i] is not equal to 63, the variable focalLengthX[i] is set equal to $\text{binToFp}(\text{sign_gvd_focal_length_x}[i], \text{exp_gvd_focal_length_x}[i], \text{man_gvd_focal_length_x}[i], v)$.

sign_gvd_focal_length_y[i] equal to 0 indicates that the sign of the focal length of the i-th camera in the vertical direction is positive. sign_gvd_focal_length_y[i] equal to 1 indicates that the sign of the focal length of the i-th camera in the vertical direction is negative.

exp_gvd_focal_length_y[i] specifies the exponent part of the focal length of the i-th camera in the vertical direction. The value of exp_gvd_focal_length_y[i] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. When exp_gvd_focal_length_y[i] is equal to 63, the value of focal length of the vertical direction is unspecified.

man_gvd_focal_length_y[i] specifies the mantissa part of the focal length of the i-th camera in the vertical direction.

The length v of the man_gvd_focal_length_y[i] syntax element is determined as follows:

- If exp_gvd_focal_length_y[i] is equal to 0, the length v is set equal to $\text{Max}(0, \text{prec_gvd_focal_length} - 30)$.
- Otherwise (exp_gvd_focal_length_y[i] is in the range of 1 to 62, inclusive), the length v is set equal to $\text{Max}(0, \text{exp_gvd_focal_length_y}[i] + \text{prec_gvd_focal_length} - 31)$.

When exp_gvd_focal_length_y[i] is not equal to 63, the variable focalLengthY[i] is set equal to $\text{binToFp}(\text{sign_gvd_focal_length_y}[i], \text{exp_gvd_focal_length_y}[i], \text{man_gvd_focal_length_y}[i], v)$.

sign_gvd_principal_point_x[i] equal to 0 indicates that the sign of the principal point of the i-th camera in the horizontal direction is positive. sign_gvd_principal_point_x[i] equal to 1 indicates that the sign of the principal point of the i-th camera in the horizontal direction is negative.

exp_gvd_principal_point_x[i] specifies the exponent part of the principal point of the i-th camera in the horizontal direction. The value of exp_gvd_principal_point_x[i] shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. When exp_gvd_principal_point_x[i] is equal to 63, the value of principal point in the horizontal direction for the i-th camera is unspecified.

man_gvd_principal_point_x[i] specifies the mantissa part of the principal point of the i-th camera in the horizontal direction. The length v of the man_gvd_principal_point_x[i] syntax element in units of bits is determined as follows:

- If exp_gvd_principal_point_x[i] is equal to 0, the length v is set equal to $\text{Max}(0, \text{prec_gvd_principal_point} - 30)$.
- Otherwise (exp_gvd_principal_point_x[i] is in the range of 1 to 62, inclusive), the length v is set equal to $\text{Max}(0, \text{exp_gvd_principal_point_x}[i] + \text{prec_gvd_principal_point} - 31)$.

When exp_gvd_principal_point_x[i] is not equal to 63, the variable principalPointX[i] is set equal to

$\text{binToFp}(\text{sign_gvd_principal_point_x}[i], \text{exp_gvd_principal_point_x}[i], \text{man_gvd_principal_point_x}[i], v)$.

sign_gvd_principal_point_y[i] equal to 0 indicates that the sign of the principal point of the i-th camera in the vertical direction is positive. **sign_gvd_principal_point_y[i]** equal to 1 indicates that the sign of the principal point of the i-th camera in the vertical direction is negative.

exp_gvd_principal_point_y[i] specifies the exponent part of the principal point of the i-th camera in the vertical direction. The value of **exp_gvd_principal_point_y[i]** shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. When **exp_gvd_principal_point_y[i]** is equal to 63, the value of principal point in the vertical direction for the i-th camera is unspecified.

man_gvd_principal_point_y[i] specifies the mantissa part of the principal point of the i-th camera in the vertical direction. The length v of the **man_gvd_principal_point_y[i]** syntax element in units of bits is determined as follows:

- If **exp_gvd_principal_point_y[i]** is equal to 0, the length v is set equal to $\text{Max}(0, \text{prec_gvd_principal_point} - 30)$.
- Otherwise (**exp_gvd_principal_point_y[i]** is in the range of 1 to 62, inclusive), the length v is set equal to $\text{Max}(0, \text{exp_gvd_principal_point_y}[i] + \text{prec_gvd_principal_point} - 31)$.

When **exp_gvd_principal_point_y[i]** is not equal to 63, the variable **principalPointY[i]** is set equal to $\text{binToFp}(\text{sign_gvd_principal_point_y}[i], \text{exp_gvd_principal_point_y}[i], \text{man_gvd_principal_point_y}[i], v)$.

sign_gvd_r[i][j][k] equal to 0 indicates that the sign of (j, k) component of the rotation matrix for the i-th camera is positive. **sign_gvd_r[i][j][k]** equal to 1 indicates that the sign of (j, k) component of the rotation matrix for the i-th camera is negative.

exp_gvd_r[i][j][k] specifies the exponent part of (j, k) component of the rotation matrix for the i-th camera. The value of **exp_gvd_r[i][j][k]** shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. When **exp_gvd_r[i][j][k]** is equal to 63, the value of rotation matrix is unspecified.

man_gvd_r[i][j][k] specifies the mantissa part of (j, k) component of the rotation matrix for the i-th camera.

The length v of the **man_gvd_r[i][j][k]** syntax element in units of bits is determined as follows:

- If **exp_gvd_r[i]** is equal to 0, the length v is set equal to $\text{Max}(0, \text{prec_gvd_rotation_param} - 30)$.
- Otherwise (**exp_gvd_r[i]** is in the range of 1 to 62, inclusive), the length v is set equal to $\text{Max}(0, \text{exp_gvd_r}[i] + \text{prec_gvd_rotation_param} - 31)$.

When **exp_gvd_r[i][j][k]** is not equal to 63, the variable **r[i][j][k]** is set equal to $\text{binToFp}(\text{sign_gvd_r}[i][j][k], \text{exp_gvd_r}[i][j][k], \text{man_gvd_r}[i][j][k], v)$.

The rotation matrix $R[i]$ for i-th camera is represented as follows:

$$\begin{bmatrix} r[i][0][0] & r[i][0][1] & r[i][0][2] \\ r[i][1][0] & r[i][1][1] & r[i][1][2] \\ r[i][2][0] & r[i][2][1] & r[i][2][2] \end{bmatrix} \quad (\text{I-316})$$

sign_gvd_t_x[i] equal to 0 indicates that the sign of the horizontal component of the translation vector for the i-th camera is positive. **sign_gvd_t_x[i]** equal to 1 indicates that the sign of the horizontal component of the translation vector for the i-th camera is negative.

exp_gvd_t_x[i] specifies the exponent part of the horizontal component of the translation vector for the i-th camera. The value of **exp_gvd_t_x[i]** shall be in the range of 0 to 62, inclusive. The value 63 is reserved for future use by ITU-T | ISO/IEC. When **exp_gvd_t_x[i]** is equal to 63, the value of the translation vector is unspecified.

man_gvd_t_x[i] specifies the mantissa part of the horizontal component of the translation vector for the i-th camera.

The length v of the **man_gvd_t_x[i]** syntax element in units of bits is determined as follows:

- If **exp_gvd_t_x[i]** is equal to 0, the length v is set equal to $\text{Max}(0, \text{prec_gvd_translation_param} - 30)$.
- Otherwise (**exp_gvd_t_x[i]** is in the range of 1 to 62, inclusive), the length v is set equal to $\text{Max}(0, \text{exp_gvd_t_x}[i] + \text{prec_gvd_translation_param} - 31)$.

When **exp_gvd_t_x[i]** is not equal to 63, the variable **tX[i]** is set equal to $\text{binToFp}(\text{sign_gvd_t_x}[i], \text{exp_gvd_t_x}[i], \text{man_gvd_t_x}[i], v)$.

When the SEI signals warp map information, syntax elements of this SEI can be used to derive a sparse set of positional correspondences between decoded pictures of different views from decoded pictures of depth layers.

min_offset_x_int, min_offset_x_frac specify the integer and the fractional part of the minimum offset for the horizontal direction of a warp map.

The variable minOffsetX is derived as follows:

$$\text{minOffsetX} = \text{min_offset_x_int} + \text{min_offset_x_frac} \div 256 \quad (\text{I-317})$$

max_offset_x_int, max_offset_x_frac specify the integer and the fractional part of the maximum offset for the horizontal direction of a warp map.

The variable maxOffsetX value is derived as follows:

$$\text{maxOffsetX} = \text{max_offset_x_int} + \text{max_offset_x_frac} \div 256 \quad (\text{I-318})$$

offset_y_present_flag equal to 1 specifies that **min_offset_y_int, min_offset_y_frac, max_offset_y_int, and max_offset_y_frac** are present. **offset_y_present_flag** equal to 0 specifies that **min_offset_y_int, min_offset_y_frac, max_offset_y_int, and max_offset_y_frac** are not present.

min_offset_y_int, min_offset_y_frac specify the integer and the fractional part of the minimum offset for the vertical direction of a warp map. When not present, the values of **min_offset_y_int** and **min_offset_y_frac** are inferred to be equal to 0.

The variable minOffsetY value is derived as follows:

$$\text{minOffsetY} = \text{min_offset_y_int} + \text{min_offset_y_frac} \div 256 \quad (\text{I-319})$$

max_offset_y_int, max_offset_y_frac specify the integer and the fractional part of the maximum offset for the vertical direction of a warp map. When not present, the values of **max_offset_y_int** and **max_offset_y_frac** are inferred to be equal to 0.

The variable maxOffsetY value is derived as follows:

$$\text{maxOffsetY} = \text{max_offset_y_int} + \text{max_offset_y_frac} \div 256 \quad (\text{I-320})$$

warp_map_size_present_flag equal to 1 specifies that a new warp map size is present, which is valid for the current and all following warp maps in output order until a new message with **warp_map_size_present_flag** equal to 1 is received or **alternative_depth_info_cancel_flag** is equal to 1. **warp_map_size_present_flag** equal to 0 specifies that the warp map size is not changed.

warp_map_width_minus2 plus 2 specifies the width of the warp map. The value of **warp_map_width_minus2** shall be in the range of 0 to (**pic_width_in_luma_samples** - 2), inclusive. The variable **warpMapWidth** is set equal to (**warp_map_width_minus2** + 2).

warp_map_height_minus2 plus 2 specifies the height of the warp map. The value of **warp_map_height_minus2** shall be in the range of 0 to (**pic_height_in_luma_samples** >> **offset_y_present_flag**) - 2, inclusive. The variable **warpMapHeight** is set equal to (**warp_map_height_minus2** + 2).

The variables **deltaX**, **deltaY**, **scaleX**, and **scaleY** are derived as follows:

$$\text{deltaX} = \text{pic_width_in_luma_samples} \div (\text{warpMapWidth} - 1) \quad (\text{I-321})$$

$$\text{deltaY} = \text{pic_height_in_luma_samples} \div (\text{warpMapHeight} - 1) \quad (\text{I-322})$$

$$\text{scaleX} = (\text{maxOffsetX} - \text{minOffsetX}) / ((1 \ll \text{BitDepth}_Y) - 1) \quad (\text{I-323})$$

$$\text{scaleY} = (\text{maxOffsetY} - \text{minOffsetY}) / ((1 \ll \text{BitDepth}_Y) - 1) \quad (\text{I-324})$$

Let **recSamples[x][y]** correspond to the reconstructed sample array S_L of a picture of a depth layer. The corresponding horizontal warp map component **w[x][y][0]** and the corresponding vertical warp map component **w[x][y][1]** for **recSamples[x][y]** are derived as follows:

```

for( x = 0; x < warpMapWidth; x++ )
    for( y = 0; y < warpMapHeight; y++ ) {
        w[x][y][0] = x * deltaX + minOffsetX + scaleX * recSamples[x][y]
        if( offset_y_present_flag )
            w[x][y][1] = y * deltaY + minOffsetY +
                        scaleY * recSamples[x][y + pic_height_in_luma_samples / 2 ]
        else
            w[x][y][1] = y * deltaY
    }

```

(I-325)

A warp map **w[x][y]** derived using the reconstructed sample array S_L of a picture included in a particular access unit and in a depth layer of a particular view is associated with the pictures included in the particular view and in the particular

access unit. The warp map specifies a sparse set of positional correspondences. These correspondences identify semantically corresponding sample locations between pictures included in the particular view and the particular AU, and the pictures included in a neighbouring view and the particular AU as follows:

- If the warp map $w[x][y]$ is associated with a picture of the leftmost view, the warp map specifies for each location $(x * \text{deltaX}, y * \text{deltaY})$ of this picture a corresponding location $(2 * w[x][y][0], 2 * w[x][y][1])$ in a picture of the closest neighbouring view on the right.
- Otherwise (the warp map $w[x][y]$ is associated with a picture of a view different to the leftmost view), the warp map specifies for each location $(x * \text{deltaX}, y * \text{deltaY})$ in this picture a corresponding location $(2 * w[x][y][0], 2 * w[x][y][1])$ in a picture of the closest neighbouring view on the left.

I.15 Video usability information

The specifications in clause G.15 apply.

Bibliography

- [1] Recommendation ITU-T H.222.0 (in force), *Information technology – Generic coding of moving pictures and associated audio information: Systems*.
ISO/IEC 13818-1(in force), *Information technology – Generic coding of moving pictures and associated audio information – Part 1: Systems*.
- [2] Recommendation ITU-T H.264 (in force), *Advanced video coding for generic audiovisual services*.
ISO/IEC 14496-10: (in force), *Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding*.
- [3] Recommendation ITU-T H.271 (in force), *Video back-channel messages for conveyance of status information and requests from a video receiver to a video sender*.
- [4] Recommendation ITU-T H.320 (in force), *Narrow-band visual telephone systems and terminal equipment*.
- [5] Recommendation ITU-T T.800 (in force), *Information technology – JPEG 2000 image coding system: Core coding system*.
ISO/IEC 15444-1 (in force), *Information technology – JPEG 2000 image coding system: Core coding system*.
- [6] Recommendation ITU-R BT. 470-6 (1998), *Conventional television systems*.
- [7] Recommendation ITU-R BT.601-7 (2011), *Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios*.
- [8] Recommendation ITU-R BT.709-6 (2015), *Parameter values for the HDTV standards for production and international programme exchange*.
- [9] Recommendation ITU-R BT.1358-0 (2007), *Studio parameters of 625 and 525 line progressive scan television systems*.
- [10] Recommendation ITU-R BT.1358-1 (2007), *Studio parameters of 625 and 525 line progressive television systems*. (Historical.)
- [11] Recommendation ITU-R BT.1361-0 (1998), *Worldwide unified colorimetry and related characteristics of future television and imaging systems*.
- [12] Recommendation ITU-R BT.1700-0 (2005), *Characteristics of composite video signals for conventional analogue television systems*.
- [13] Recommendation ITU-R BT.1886-0 (2011), *Reference electro-optical transfer function for flat panel displays used in HDTV studio production*.
- [14] Recommendation ITU-R BT.2020-2 (2015), *Parameter values for ultra-high definition television systems for production and international programme exchange*.
- [15] Recommendation ITU-R BT.2035 (2013), *A reference viewing environment for evaluation of HDTV program material or completed programmes*.
- [16] Recommendation ITU-R BT.2100-2 (2018), *Image parameter values for high dynamic range television for use in production and international programme exchange*.
- [17] ANSI/CTA 861-G (2016), *A DTV Profile for Uncompressed High Speed Digital Interfaces*.
- [18] ARIB STD-B67 (2015), *Essential Parameter Values for the Extended Image Dynamic Range Television (EIDRTV) System for Programme Production*.
- [19] ATSC A/341 (2019), *Video – HEVC*.
- [20] CIE 15 (in force), *Colorimetry*.
- [21] CEA 861.3 (2015), *HDR Static Metadata Extensions*.
- [22] EBU Tech. 3213-E (1975), *EBU Standard for Chromaticity Tolerances for Studio Monitors*.
- [23] IEC 60559:1989, *Binary floating-point arithmetic for microprocessor systems*.
- [24] IEC 61966-2-1 (in force), *Multimedia systems and equipment – Colour measurement and management – Part 2-1: Colour management – Default RGB colour space – sRGB*.
- [25] IEC 61966-2-4 (in force), *Multimedia systems and equipment – Colour measurement and management – Part 2-4: Colour management – Extended-gamut YCC colour space for video applications – xvYCC*.

- [26] Internet Engineering Task Force, RFC 3550, Standard 64 (in force), *RTP: A Transport Protocol for Real-Time Applications*.
- [27] ISO 11664-3 (in force), *Colorimetry – Part 3: CIE tristimulus values*.
- [28] ISO/IEC 14496-12: *Information technology – Coding of audio-visual objects – Part 12: ISO base media file format*.
- [29] SMPTE EG 432-1 (2010), *Digital Source Processing – Color Processing for D-Cinema*.
- [30] SMPTE RDD 5 (2006), *Film Grain Technology – Specifications for H.264/MPEG-4 AVC Bitstreams*.
- [31] SMPTE RP 177 (1993), *Derivation of Basic Television Color Equations*.
- [32] SMPTE RP 431-2 (2011), *D-Cinema Quality – Reference Projector and Environment, Annex C*.
- [33] SMPTE RP 2050-1 (2012), *4:2:2 / 4:2:0 Format Conversion Minimizing Color Difference Signal Degradation in Concatenated Operations – Filtering*.
- [34] SMPTE ST 12-1 (2014), *Time and Control Code*.
- [35] SMPTE ST 170 (2004), *Television – Composite Analog Video Signal – NTSC for Studio Applications*.
- [36] SMPTE ST 240 (1999), *Television – 1125-Line High-Definition Production Systems – Signal Parameters*.
- [37] SMPTE ST 428-1 (2006), *D-Cinema Distribution Master (DCDM) – Image Characteristics*.
- [38] SMPTE ST 2084 (2014), *High Dynamic Range Electro-Optical Transfer Function of Mastering Reference Displays*.
- [39] SMPTE ST 2085 (2015), *$Y'D'zD'_x$ Color-Difference Computations for High Dynamic Range $XY'Z'$ Signals*.
- [40] SMPTE ST 2086 (2018), *Mastering Display Color Volume Metadata supporting High Luminance and Wide Color Gamut Images*.
- [41] SMPTE ST 2113 (2019), *Colorimetry of P3 Color Spaces*.
- [42] United States Federal Communications Commission (2003), *Title 47 Code of Federal Regulations 73.682 (a) (20)*.
- [43] United States National Television System Committee (1953), *Recommendation for transmission standards for colour television*.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	Tariff and accounting principles and international telecommunication/ICT economic and policy issues
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Environment and ICTs, climate change, e-waste, energy efficiency; construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling, and associated measurements and tests
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	Global information infrastructure, Internet protocol aspects, next-generation networks, Internet of Things and smart cities
Series Z	Languages and general software aspects for telecommunication systems

EXHIBIT 4



US007532808B2

(12) **United States Patent**
Lainema

(10) **Patent No.:** **US 7,532,808 B2**
(45) **Date of Patent:** **May 12, 2009**

(54) **METHOD FOR CODING MOTION IN A VIDEO SEQUENCE**

(75) Inventor: **Jani Lainema**, Irving, TX (US)

(73) Assignee: **Nokia Corporation**, Espoo (FI)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1003 days.

(21) Appl. No.: **10/390,549**

(22) Filed: **Mar. 14, 2003**

(65) **Prior Publication Data**

US 2003/0202594 A1 Oct. 30, 2003

Related U.S. Application Data

(60) Provisional application No. 60/365,072, filed on Mar. 15, 2002.

(51) **Int. Cl.**
H04N 5/91 (2006.01)

(52) **U.S. Cl.** **386/111**; 386/112

(58) **Field of Classification Search** 386/68, 386/111, 112, 95; 348/466, 699; 375/240.15
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,148,272 A	9/1992	Acampora et al.	358/133
5,191,436 A	3/1993	Yonemitsu	358/335
5,442,400 A	8/1995	Sun et al.	348/402
5,701,164 A	12/1997	Kato	348/699
6,683,987 B1 *	1/2004	Sugahara	382/235
7,200,275 B2 *	4/2007	Srinivasan et al.	382/239

OTHER PUBLICATIONS

"Global Motion Vector Coding (GMVC)"; Shijun Sun et al.; ITU—Telecommunications Standardization Sector, Video Coding Experts Group (VCEG); Meeting: Pattaya, Thailand, Dec. 4-7, 2001; pp. 1-6.

"Joint Model Number 1 (JM-1)"; Doc. JVT-A003; Joint Video Team of ISO/IEC and ITU-T VCEG; Jan. 2002; pp. 1-79.

Acta of Zhongshan University, vol. 40, No. 2; L. Hongmei et al.; "An Improved Multiresolution Motion Estimation Algorithm"; pp. 34-37; Mar. 2001.

ITU Telecommunications Standardization Sector, Doc. VCEG-N77; S. Sun et al.; "Motion Vector Coding with Global Motion Parameters"; pp. 1-11; Fourteenth Meeting: Santa Barbara, CA, USA, Sep. 24-28, 2001.

(Continued)

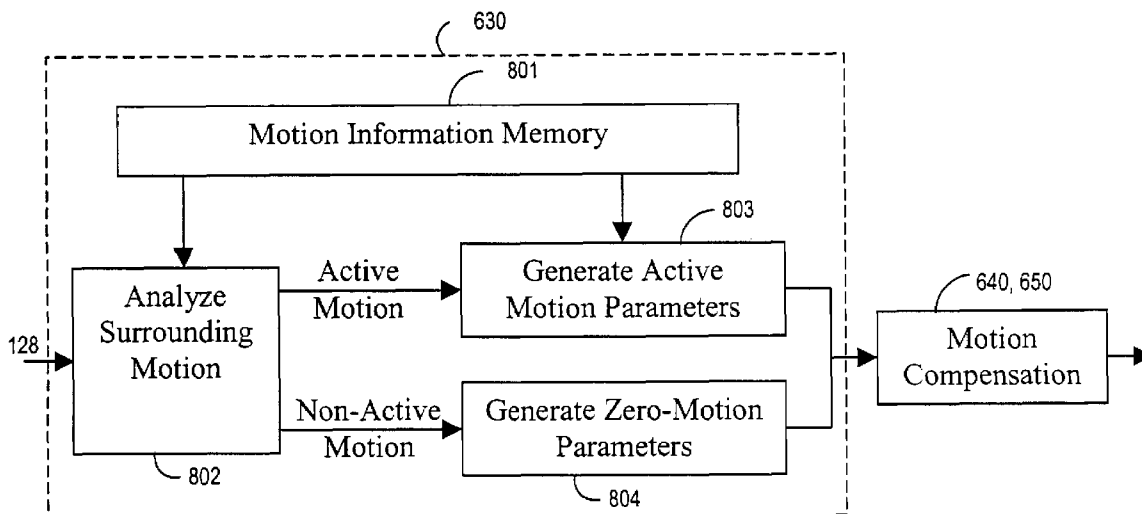
Primary Examiner—Huy T Nguyen

(74) *Attorney, Agent, or Firm*—Ware, Fressola, Van Der Sluys & Adolphson, LLP

(57) **ABSTRACT**

A method of motion-compensated video encoding that enables a video sequence with a global motion component to be encoded in an efficient manner. A video encoder is arranged to assign macroblocks to be coded to specific coding modes including a skip mode, which is used to indicate one of two possible types of macroblock motion: a) zero motion, or b) global or regional motion. As each macroblock is encoded, a previously encoded region surrounding the macroblock is examined and the characteristics of motion in that region determined. With the skip mode, the macroblock to be coded and a motion vector describing the global motion or regional motion is associated with the macroblock if the motion in the region is characteristic of global motion or regional motion. If the region exhibits an insignificant level of motion, a zero valued motion vector is associated with the macroblock.

65 Claims, 10 Drawing Sheets



OTHER PUBLICATIONS

ITU Telecommunications Standardization Sector, Doc. VCEG-N16;
S. Sun et al; "Core Experiment description: Motion Vector Coding
with Global Motion Parameters"; pp. 1-6; Fourteenth Meeting: Santa
Barbara, CA, USA, Sep. 24-28, 2001.

Joint Photography Expert Group Conference, Crowborough JPEG
Forum Ltd, GB, Specialists Group on Coding for Visual Telephony
Joint Photographic Expert Group; "Description of Ref. Model 8
(RM8)"; pp. 1-72; Jun. 9, 1989.

* cited by examiner

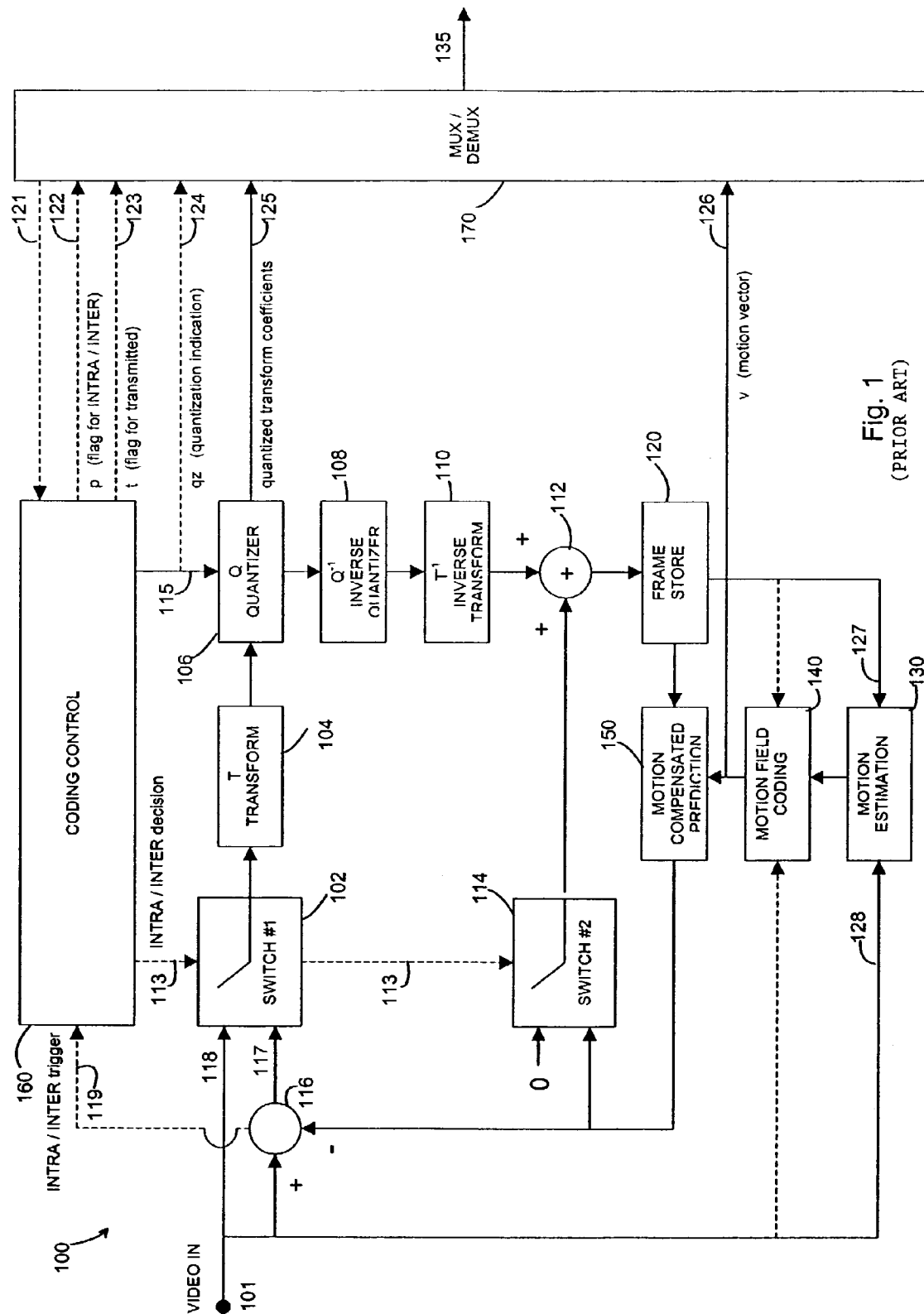


Fig. 1
(PRIOR ART)

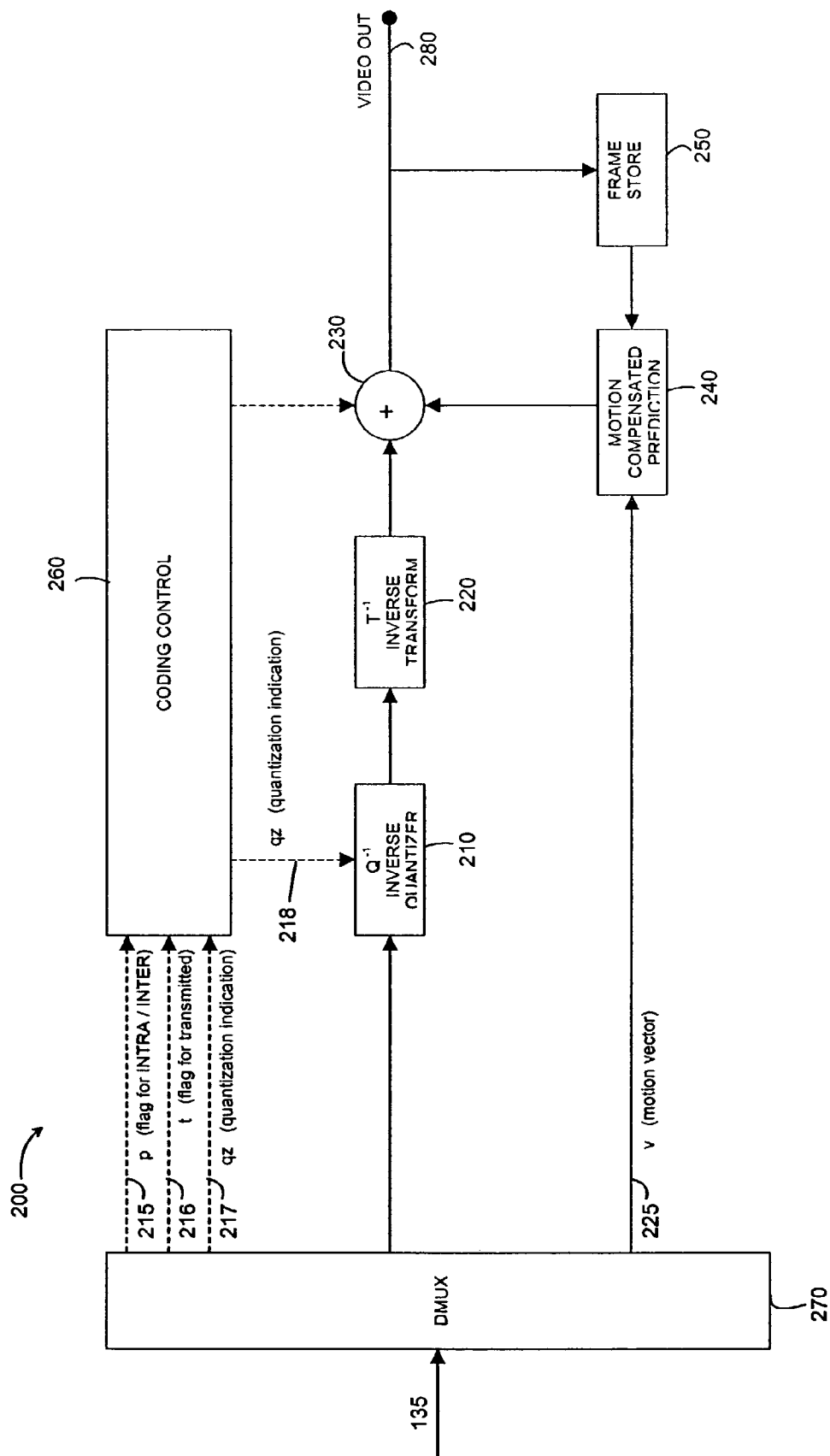


Fig. 2
(PRIOR ART)

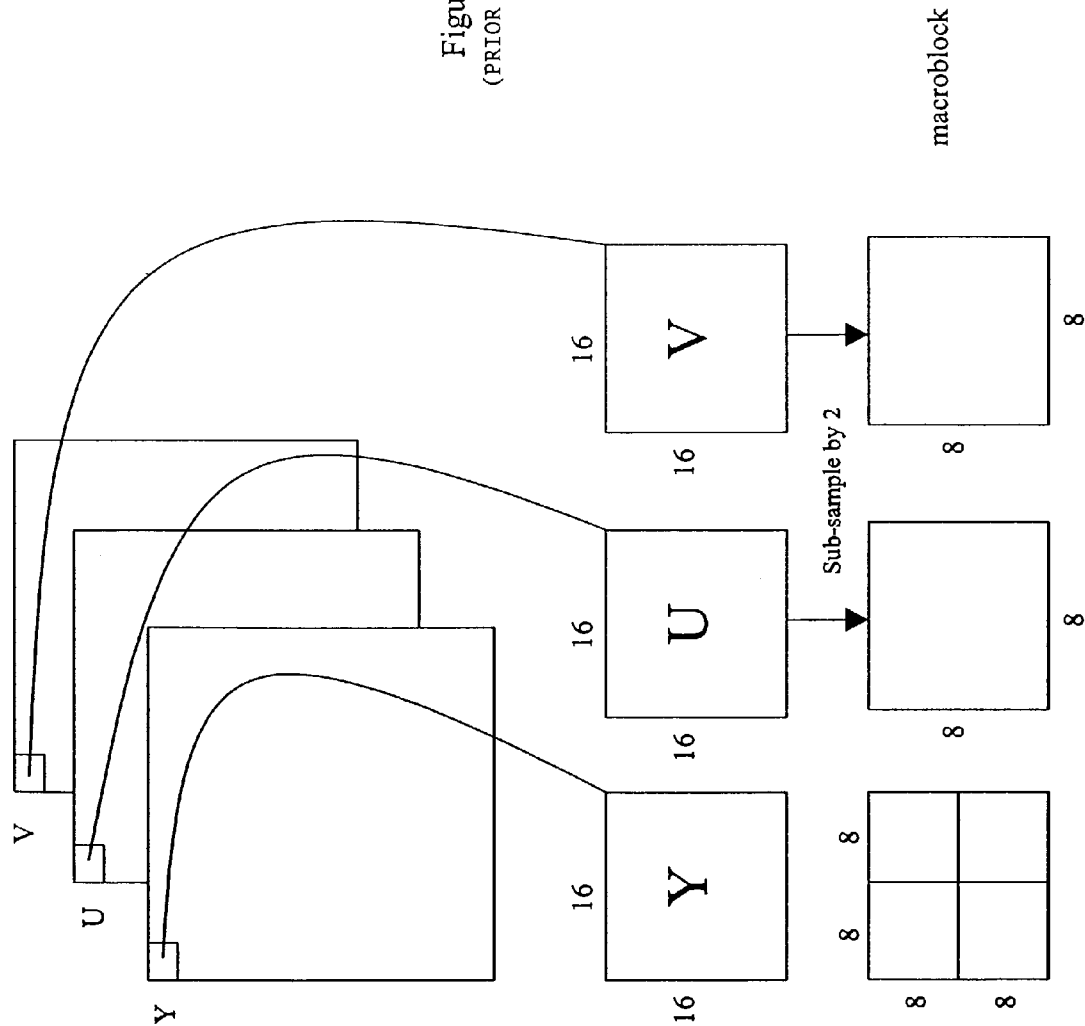


Figure 3
(PRIOR ART)

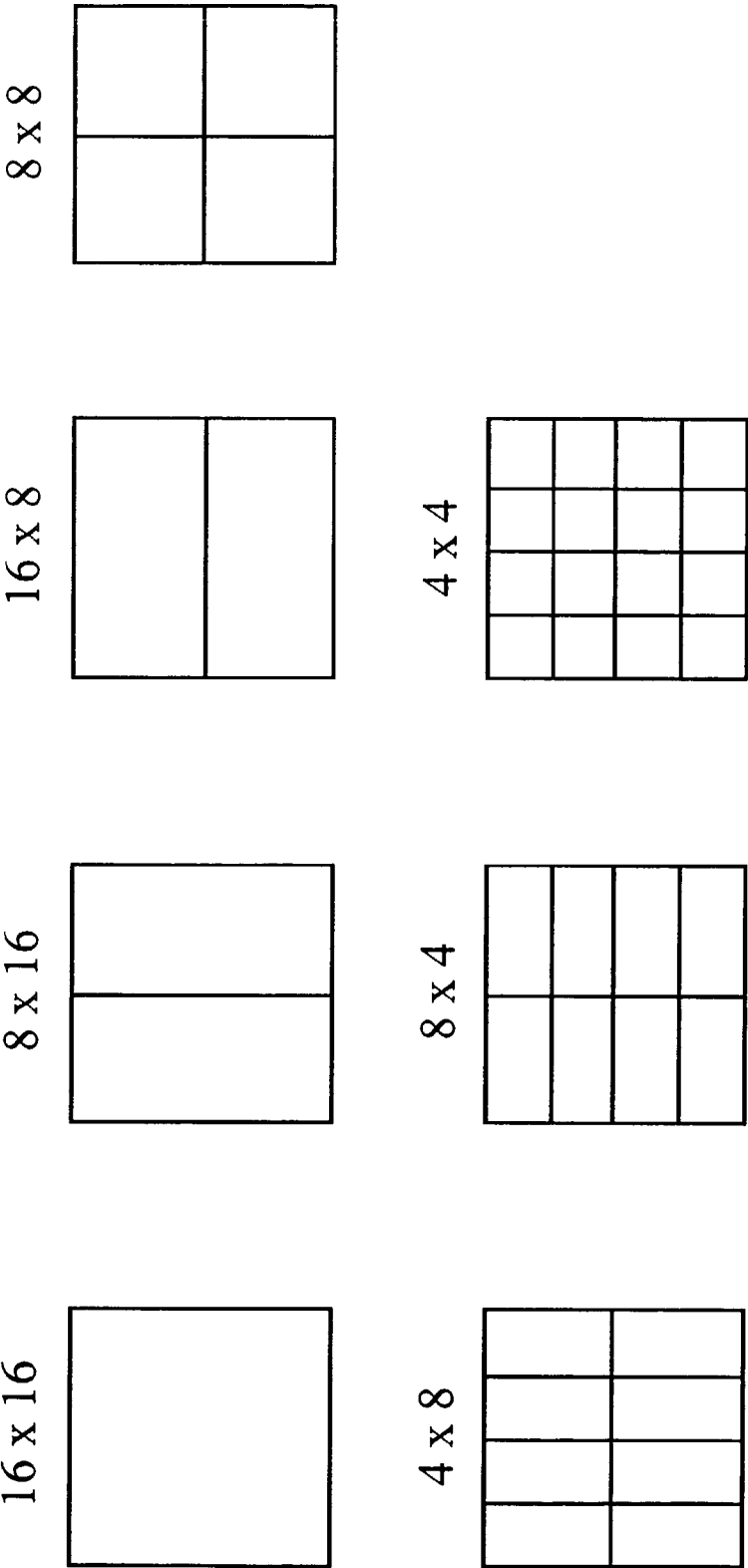


FIG. 4

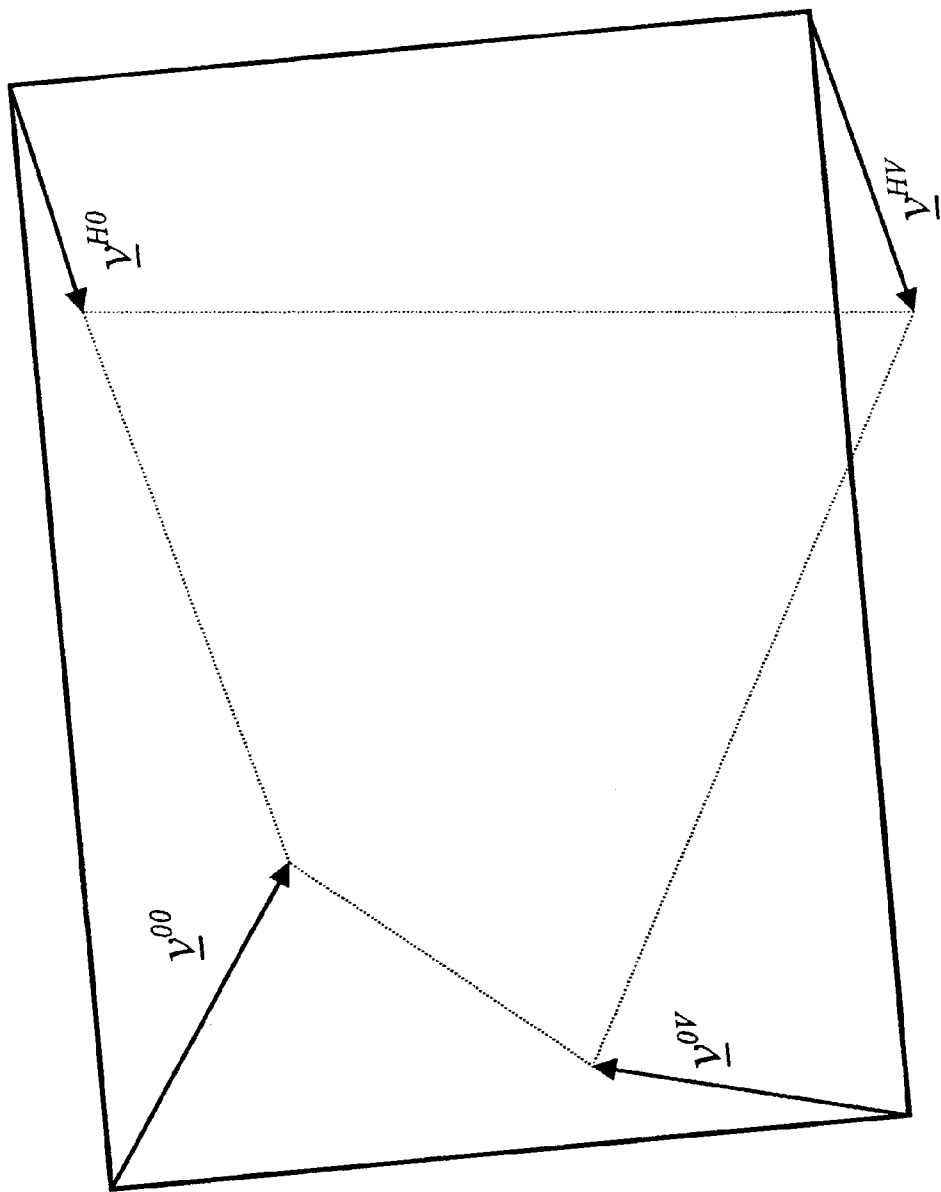


FIG. 5

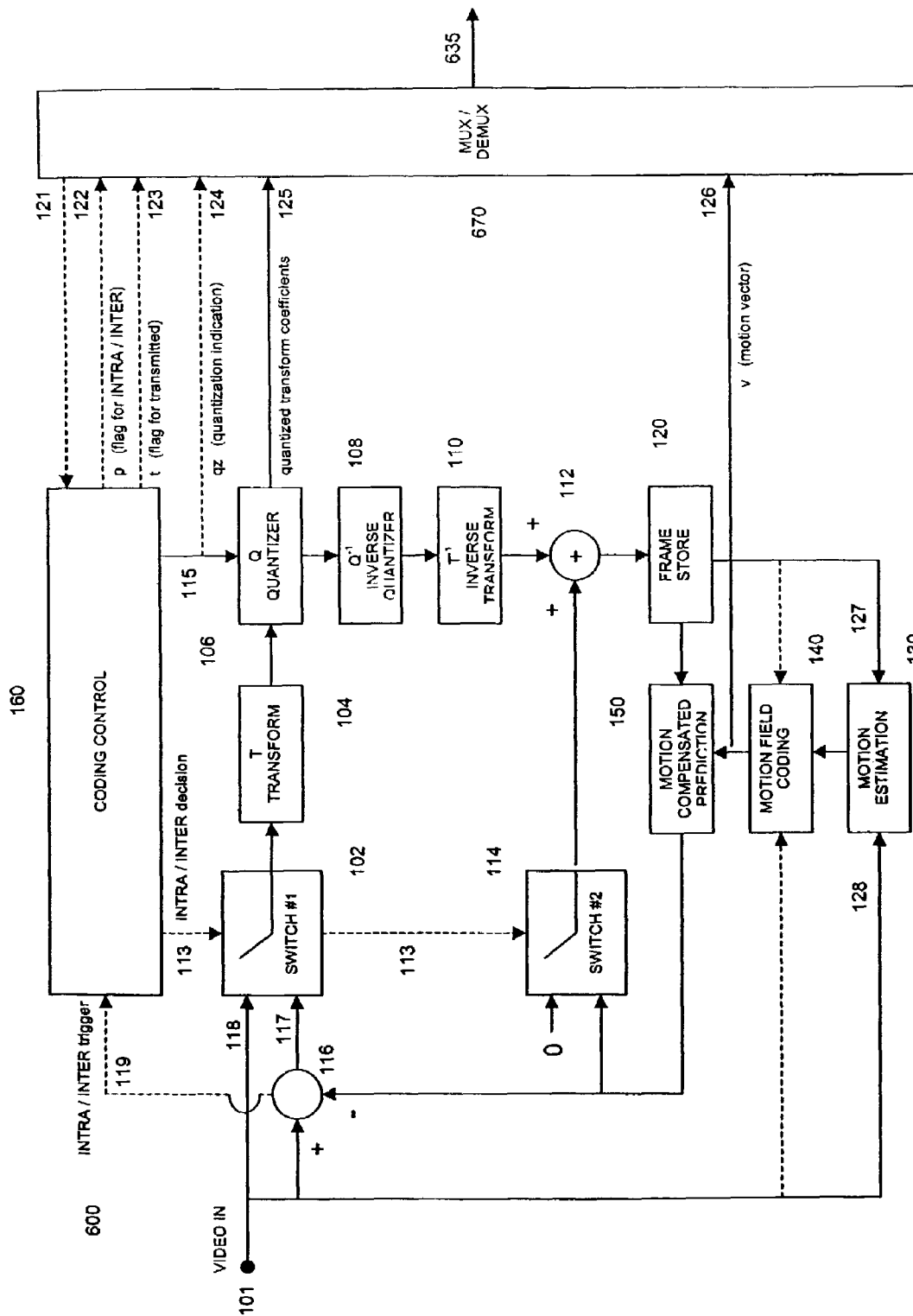


FIG. 6

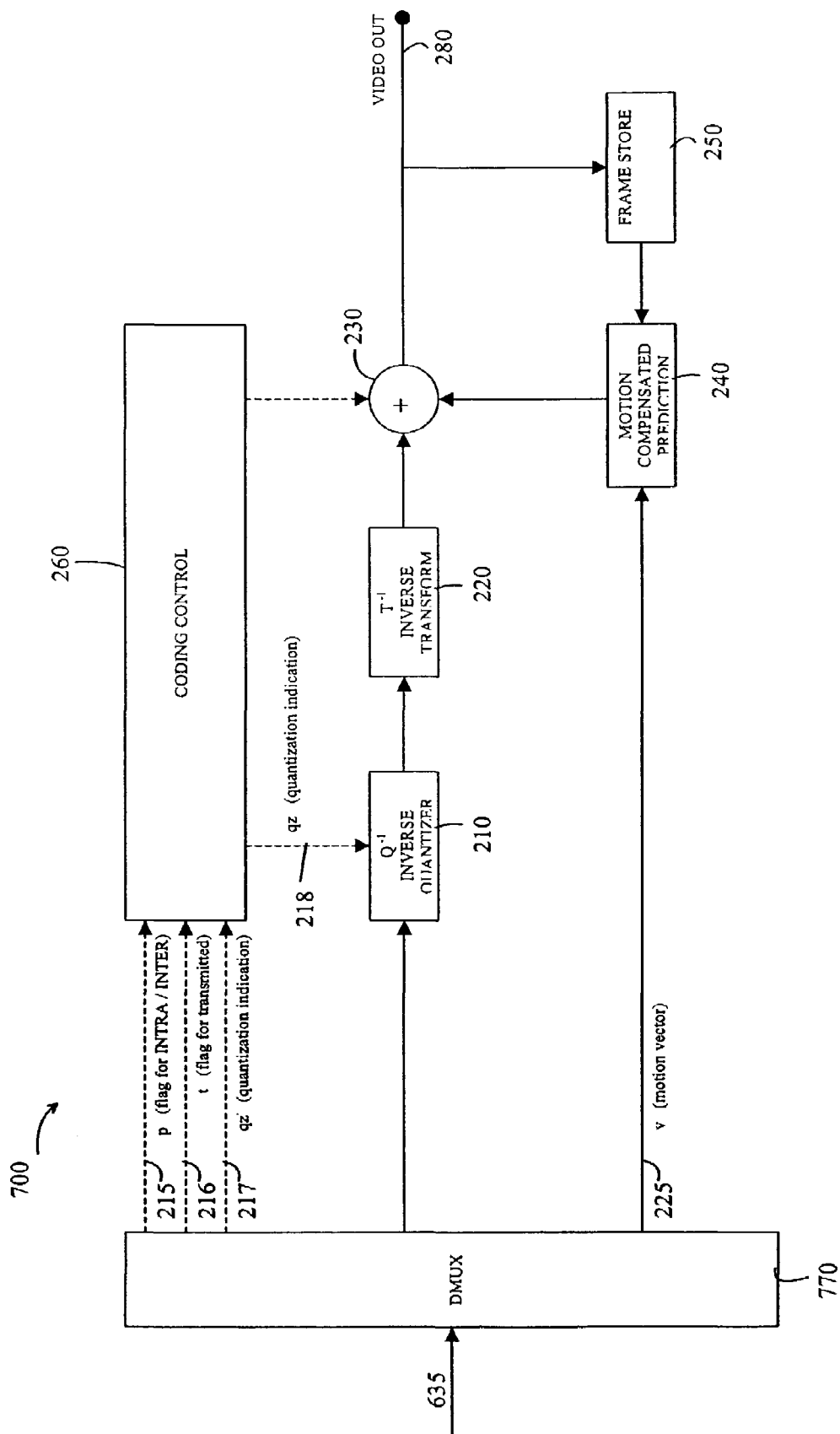


Fig. 7

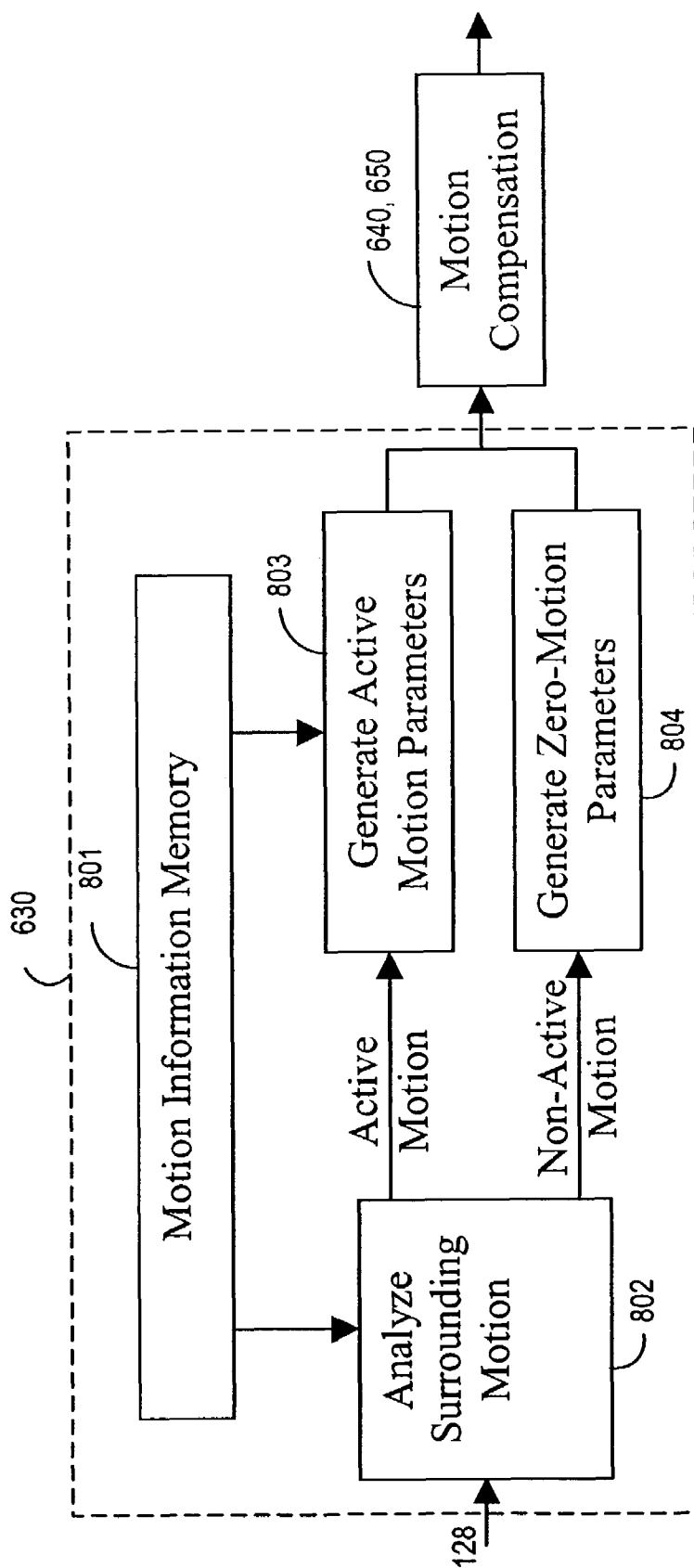


FIG. 8

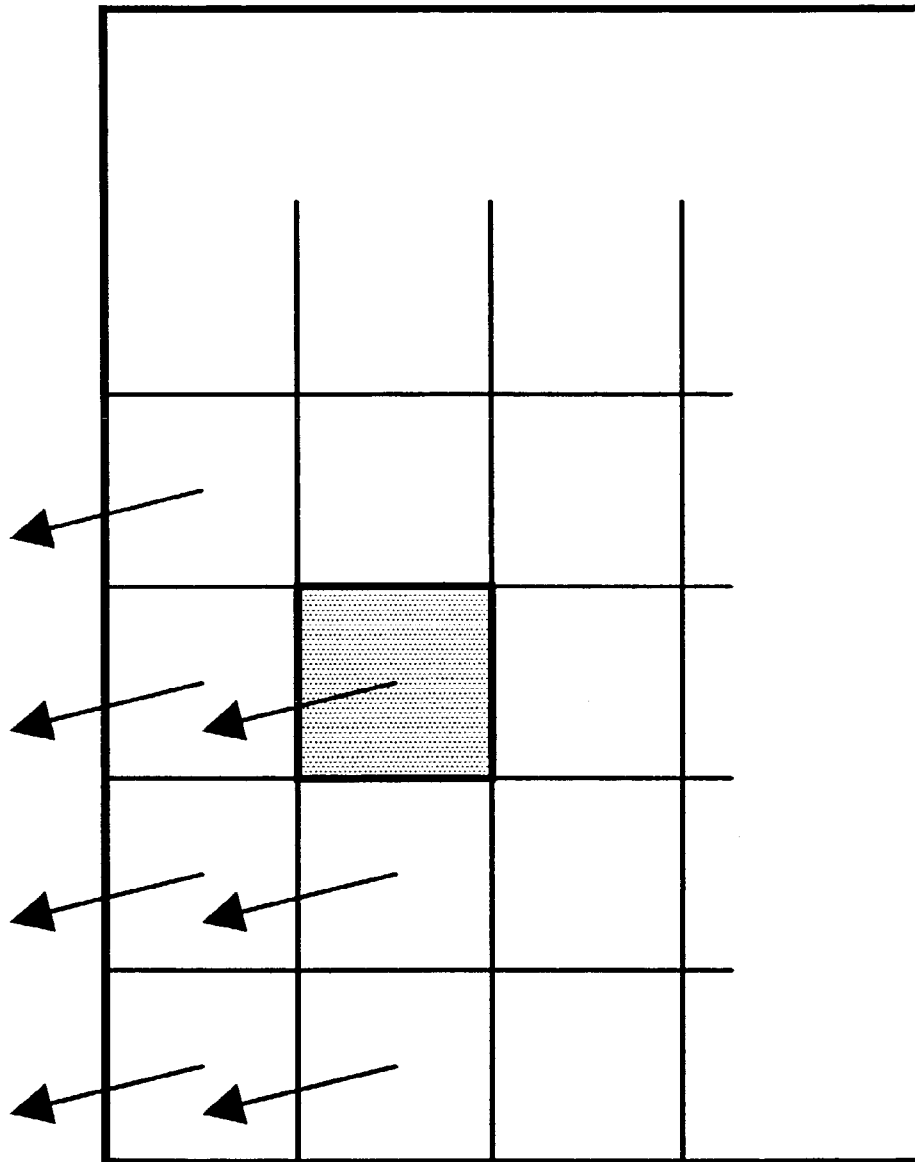


FIG. 9

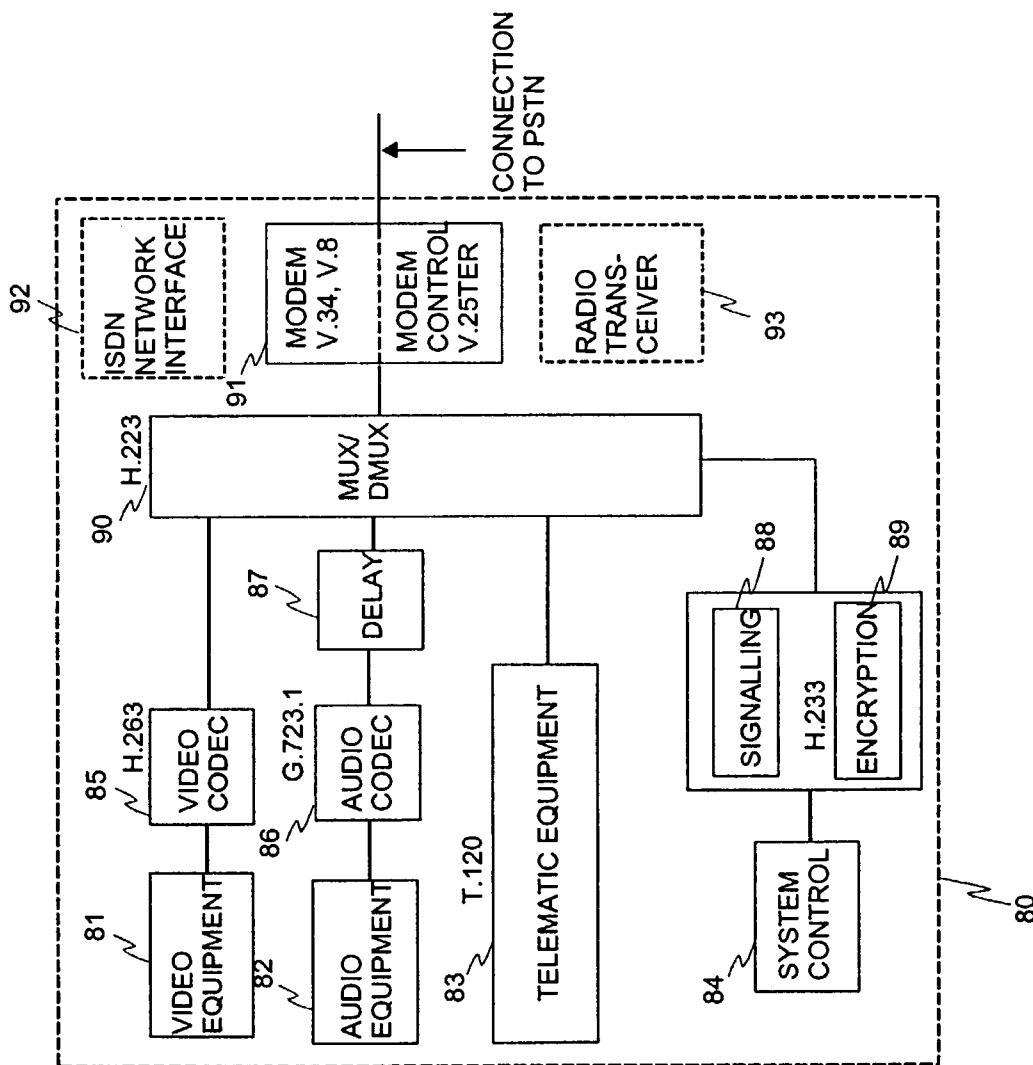


Fig. 10

1

METHOD FOR CODING MOTION IN A VIDEO SEQUENCE

This application claims the benefit of U.S. Provisional Application No. 60/365,072 filed Mar. 15, 2002.

FIELD OF THE INVENTION

The invention relates generally to communication systems and more particularly to motion compensation in video coding.

BACKGROUND OF THE INVENTION

A digital video sequence, like an ordinary motion picture recorded on film, comprises a sequence of still images, the illusion of motion being created by displaying consecutive images of the sequence one after the other at a relatively fast rate, typically 15 to 30 frames per second. Because of the relatively fast frame display rate, images in consecutive frames tend to be quite similar and thus contain a considerable amount of redundant information. For example, a typical scene may comprise some stationary elements, such as background scenery, and some moving areas, which may take many different forms, for example the face of a newsreader, moving traffic and so on. Alternatively, or additionally, so-called "global motion" may be present in the video sequence, for example due to translation, panning or zooming of the camera recording the scene. However, in many cases, the overall change between one video frame and the next is rather small.

Each frame of an uncompressed digital video sequence comprises an array of image pixels. For example, in a commonly used digital video format, known as the Quarter Common Interchange Format (QCIF), a frame comprises an array of 176×144 pixels, in which case each frame has 25,344 pixels. In turn, each pixel is represented by a certain number of bits, which carry information about the luminance and/or color content of the region of the image corresponding to the pixel. Commonly, a so-called YUV color model is used to represent the luminance and chrominance content of the image. The luminance, or Y, component represents the intensity (brightness) of the image, while the color content of the image is represented by two chrominance or color difference components, labelled U and V.

Color models based on a luminance/chrominance representation of image content provide certain advantages compared with color models that are based on a representation involving primary colors (that is Red, Green and Blue, RGB). The human visual system is more sensitive to intensity variations than it is to color variations and YUV color models exploit this property by using a lower spatial resolution for the chrominance components (U, V) than for the luminance component (Y). In this way, the amount of information needed to code the color information in an image can be reduced with an acceptable reduction in image quality.

The lower spatial resolution of the chrominance components is usually attained by spatial sub-sampling. Typically, each frame of a video sequence is divided into so-called "macroblocks", which comprise luminance (Y) information and associated (spatially sub-sampled) chrominance (U, V) information. FIG. 3 illustrates one way in which macroblocks can be formed. FIG. 3a shows a frame of a video sequence represented using a YUV color model, each component having the same spatial resolution. Macroblocks are formed by representing a region of 16×16 image pixels in the original image (FIG. 3b) as four blocks of luminance information,

2

each luminance block comprising an 8×8 array of luminance (Y) values and two spatially corresponding chrominance components (U and V) which are sub-sampled by a factor of two in the horizontal and vertical directions to yield corresponding arrays of 8×8 chrominance (U, V) values (see FIG. 3c).

A QCIF image comprises 11×9 macroblocks. If the luminance blocks and chrominance blocks are represented with 8 bit resolution (that is by numbers in the range 0 to 255), the total number of bits required per macroblock is $(16 \times 16 \times 8) + 2 \times (8 \times 8 \times 8) = 3072$ bits. The number of bits needed to represent a video frame in QCIF format is thus $99 \times 3072 = 304,128$ bits. This means that the amount of data required to transmit/record/display an uncompressed video sequence in QCIF format, represented using a YUV color model, at a rate of 30 frames per second, is more than 9 Mbps (million bits per second). This is an extremely high data rate and is impractical for use in video recording, transmission and display applications because of the very large storage capacity, transmission channel capacity and hardware performance required.

If video data is to be transmitted in real-time over a fixed line network such as an ISDN (Integrated Services Digital Network) or a conventional PSTN (Public Switched Telephone Network), the available data transmission bandwidth is typically of the order of 64 kbits/s. In mobile videotelephony, where transmission takes place at least in part over a radio communications link, the available bandwidth can be as low as 20 kbits/s. This means that a significant reduction in the amount of information used to represent video data must be achieved in order to enable transmission of digital video sequences over low bandwidth communication networks. For this reason, video compression techniques have been developed which reduce the amount of information transmitted while retaining an acceptable image quality.

Video compression methods are based on reducing the redundant and perceptually irrelevant parts of video sequences. The redundancy in video sequences can be categorised into spatial, temporal and spectral redundancy. "Spatial redundancy" is the term used to describe the correlation (similarity) between neighbouring pixels within a frame. The term "temporal redundancy" expresses the fact that objects appearing in one frame of a sequence are likely to appear in subsequent frames, while "spectral redundancy" refers to the correlation between different color components of the same image.

Sufficiently efficient compression cannot usually be achieved by simply reducing the various forms of redundancy in a given sequence of images. Thus, most current video encoders also reduce the quality of those parts of the video sequence which are subjectively the least important. In addition, the redundancy of the compressed video bit-stream itself is reduced by means of efficient loss-less encoding. Generally, this is achieved using a technique known as entropy coding.

There is often a significant amount of spatial redundancy between the pixels that make up each frame of a digital video sequence. In other words, the value of any pixel within a frame of the sequence is substantially the same as the value of other pixels in its immediate vicinity. Typically, video coding systems reduce spatial redundancy using a technique known as "block-based transform coding", in which a mathematical transformation, such as a two-dimensional Discrete Cosine Transform (DCT), is applied to blocks of image pixels. This transforms the image data from a representation comprising pixel values to a form comprising a set of coefficient values representative of spatial frequency components significantly

reducing spatial redundancy and thereby producing a more compact representation of the image data.

Frames of a video sequence which are compressed using block-based transform coding, without reference to any other frame within the sequence, are referred to as INTRA-coded or I-frames. Additionally, and where possible, blocks of INTRA-coded frames are predicted from previously coded blocks within the same frame. This technique, known as INTRA-prediction, has the effect of further reducing the amount of data required to represent an INTRA-coded frame.

Generally, video coding systems not only reduce the spatial redundancy within individual frames of a video sequence, but also make use of a technique known as "motion-compensated prediction", to reduce the temporal redundancy in the sequence. Using motion-compensated prediction, the image content of some (often many) frames in a digital video sequence is "predicted" from one or more other frames in the sequence, known as "reference" frames. Prediction of image content is achieved by tracking the motion of objects or regions of an image between a frame to be coded (compressed) and the reference frame(s) using "motion vectors". In general, the reference frame(s) may precede the frame to be coded or may follow it in the video sequence. As in the case of INTRA-coding, motion compensated prediction of a video frame is typically performed macroblock-by-macroblock.

Frames of a video sequence which are compressed using motion-compensated prediction are generally referred to as INTER-coded or P-frames. Motion-compensated prediction alone rarely provides a sufficiently precise representation of the image content of a video frame and therefore it is typically necessary to provide a so-called "prediction error" (PE) frame with each INTER-coded frame. The prediction error frame represents the difference between a decoded version of the INTER-coded frame and the image content of the frame to be coded. More specifically, the prediction error frame comprises values that represent the difference between pixel values in the frame to be coded and corresponding reconstructed pixel values formed on the basis of a predicted version of the frame in question. Consequently, the prediction error frame has characteristics similar to a still image and block-based transform coding can be applied in order to reduce its spatial redundancy and hence the amount of data (number of bits) required to represent it.

In order to illustrate the operation of a generic video coding system in greater detail, reference will now be made to the exemplary video encoder and video decoder illustrated in FIGS. 1 and 2 of the accompanying drawings. The video encoder **100** of FIG. 1 employs a combination of INTRA- and INTER-coding to produce a compressed (encoded) video bit-stream and decoder **200** of FIG. 2 is arranged to receive and decode the video bit-stream produced by encoder **100** in order to produce a reconstructed video sequence. Throughout the following description it will be assumed that the luminance component of a macroblock comprises 16x16 pixels arranged as an array of 4, 8x8 blocks, and that the associated chrominance components are spatially sub-sampled by a factor of two in the horizontal and vertical directions to form 8x8 blocks, as depicted in FIG. 3. Extension of the description to other block sizes and other sub-sampling schemes will be apparent to those of ordinary skill in the art.

The video encoder **100** comprises an input **101** for receiving a digital video signal from a camera or other video source (not shown). It also comprises a transformation unit **104** which is arranged to perform a block-based discrete cosine transform (DCT), a quantizer **106**, an inverse quantizer **108**, an inverse transformation unit **110**, arranged to perform an inverse block-based discrete cosine transform (IDCT), com-

biners **112** and **116**, and a frame store **120**. The encoder further comprises a motion estimator **130**, a motion field coder **140** and a motion compensated predictor **150**. Switches **102** and **114** are operated co-operatively by control manager **160** to switch the encoder between an INTRA-mode of video encoding and an INTER-mode of video encoding. The encoder **100** also comprises a video multiplex coder **170** which forms a single bit-stream from the various types of information produced by the encoder **100** for further transmission to a remote receiving terminal or, for example, for storage on a mass storage medium, such as a computer hard drive (not shown).

Encoder **100** operates as follows. Each frame of uncompressed video provided from the video source to input **101** is received and processed macroblock by macroblock, preferably in raster-scan order. When the encoding of a new video sequence starts, the first frame to be encoded is encoded as an INTRA-coded frame. Subsequently, the encoder is programmed to code each frame in INTER-coded format, unless one of the following conditions is met: 1) it is judged that the current macroblock of the frame being coded is so dissimilar from the pixel values in the reference frame used in its prediction that excessive prediction error information is produced, in which case the current macroblock is coded in INTRA-coded format; 2) a predefined INTRA frame repetition interval has expired; or 3) feedback is received from a receiving terminal indicating a request for a frame to be provided in INTRA-coded format.

The occurrence of condition 1) is detected by monitoring the output of the combiner **116**. The combiner **116** forms a difference between the current macroblock of the frame being coded and its prediction, produced in the motion compensated prediction block **150**. If a measure of this difference (for example a sum of absolute differences of pixel values) exceeds a predetermined threshold, the combiner **116** informs the control manager **160** via a control line **119** and the control manager **160** operates the switches **102** and **114** via control line **113** so as to switch the encoder **100** into INTRA-coding mode. In this way, a frame which is otherwise encoded in INTER-coded format may comprise INTRA-coded macroblocks. Occurrence of condition 2) is monitored by means of a timer or frame counter implemented in the control manager **160**, in such a way that if the timer expires, or the frame counter reaches a predetermined number of frames, the control manager **160** operates the switches **102** and **114** via control line **113** to switch the encoder into INTRA-coding mode. Condition 3) is triggered if the control manager **160** receives a feedback signal from, for example, a receiving terminal, via control line **121** indicating that an INTRA frame refresh is required by the receiving terminal. Such a condition may arise, for example, if a previously transmitted frame is badly corrupted by interference during its transmission, rendering it impossible to decode at the receiver. In this situation, the receiving decoder issues a request for the next frame to be encoded in INTRA-coded format, thus re-initialising the coding sequence.

Operation of the encoder **100** in INTRA-coding mode will now be described. In INTRA-coding mode, the control manager **160** operates the switch **102** to accept video input from input line **118**. The video signal input is received macroblock by macroblock from input **101** via the input line **118**. As they are received, the blocks of luminance and chrominance values which make up the macroblock are passed to the DCT transformation block **104**, which performs a 2-dimensional discrete cosine transform on each block of values, producing a 2-dimensional array of DCT coefficients for each block. DCT transformation block **104** produces an array of coefficient

values for each block, the number of coefficient values corresponding to the dimensions of the blocks which make up the macroblock (in this case 8x8). The DCT coefficients for each block are passed to the quantizer **106**, where they are quantized using a quantization parameter QP. Selection of the quantization parameter QP is controlled by the control manager **160** via control line **115**.

The array of quantized DCT coefficients for each block is then passed from the quantizer **106** to the video multiplex coder **170**, as indicated by line **125** in FIG. 1. The video multiplex coder **170** orders the quantized transform coefficients for each block using a zigzag scanning procedure, thereby converting the two-dimensional array of quantized transform coefficients into a one-dimensional array. Each non-zero valued quantized coefficient in the one dimensional array is then represented as a pair of values, referred to as level and run, where level is the value of the quantized coefficient and run is the number of consecutive zero-valued coefficients preceding the coefficient in question. The run and level values are further compressed in the video multiplex coder **170** using entropy coding, for example, variable length coding (VLC), or arithmetic coding.

Once the run and level values have been entropy coded using an appropriate method, the video multiplex coder **170** further combines them with control information, also entropy coded using a method appropriate for the kind of information in question, to form a single compressed bit-stream of coded image information **135**. It should be noted that while entropy coding has been described in connection with operations performed by the video multiplex coder **170**, in alternative implementations a separate entropy coding unit may be provided.

A locally decoded version of the macroblock is also formed in the encoder **100**. This is done by passing the quantized transform coefficients for each block, output by quantizer **106**, through inverse quantizer **108** and applying an inverse DCT transform in inverse transformation block **110**. In this way a reconstructed array of pixel values is constructed for each block of the macroblock. The resulting decoded image data is input to combiner **112**. In INTRA-coding mode, switch **114** is set so that the input to the combiner **112** via switch **114** is zero. In this way, the operation performed by combiner **112** is equivalent to passing the decoded image data unaltered.

As subsequent macroblocks of the current frame are received and undergo the previously described encoding and local decoding steps in blocks **104**, **106**, **108**, **110** and **112**, a decoded version of the INTRA-coded frame is built up in frame store **120**. When the last macroblock of the current frame has been INTRA-coded and subsequently decoded, the frame store **120** contains a completely decoded frame, available for use as a motion prediction reference frame in coding a subsequently received video frame in INTER-coded format.

Operation of the encoder **100** in INTER-coding mode will now be described. In INTER-coding mode, the control manager **160** operates switch **102** to receive its input from line **117**, which comprises the output of combiner **116**. The combiner **116** receives the video input signal macroblock by macroblock from input **101**. As combiner **116** receives the blocks of luminance and chrominance values which make up the macroblock, it forms corresponding blocks of prediction error information. The prediction error information represents the difference between the block in question and its prediction, produced in motion compensated prediction block **150**. More specifically, the prediction error information for each block of the macroblock comprises a two-dimensional array of values, each of which represents the difference

between a pixel value in the block of luminance or chrominance information being coded and a decoded pixel value obtained by forming a motion-compensated prediction for the block, according to the procedure to be described below. Thus, in the exemplary video coding system considered here where each macroblock comprises, for example, an assembly of 8x8 blocks comprising luminance and chrominance values, the prediction error information for each block of the macroblock similarly comprises an 8x8 array of prediction error values.

The prediction error information for each block of the macroblock is passed to DCT transformation block **104**, which performs a two-dimensional discrete cosine transform on each block of prediction error values to produce a two-dimensional array of DCT transform coefficients for each block. DCT transformation block **104** produces an array of coefficient values for each prediction error block, the number of coefficient values corresponding to the dimensions of the blocks which make up the macroblock (in this case 8x8). The transform coefficients derived from each prediction error block are passed to quantizer **106** where they are quantized using a quantization parameter QP, in a manner analogous to that described above in connection with operation of the encoder in INTRA-coding mode. As before, selection of the quantization parameter QP is controlled by the control manager **160** via control line **115**.

The quantized DCT coefficients representing the prediction error information for each block of the macroblock are passed from quantizer **106** to video multiplex coder **170**, as indicated by line **125** in FIG. 1. As in INTRA-coding mode, the video multiplex coder **170** orders the transform coefficients for each prediction error block using a certain zigzag scanning procedure and then represents each non-zero valued quantized coefficient as a run-level pair. It further compresses the run-level pairs using entropy coding, in a manner analogous to that described above in connection with INTRA-coding mode. Video multiplex coder **170** also receives motion vector information (described in the following) from motion field coding block **140** via line **126** and control information from control manager **160**. It entropy codes the motion vector information and control information and forms a single bit-stream of coded image information, **135** comprising the entropy coded motion vector, prediction error and control information.

The quantized DCT coefficients representing the prediction error information for each block of the macroblock are also passed from quantizer **106** to inverse quantizer **108**. Here they are inverse quantized and the resulting blocks of inverse quantized DCT coefficients are applied to inverse DCT transform block **110**, where they undergo inverse DCT transformation to produce locally decoded blocks of prediction error values. The locally decoded blocks of prediction error values are then input to combiner **112**. In INTER-coding mode, switch **114** is set so that the combiner **112** also receives predicted pixel values for each block of the macroblock, generated by motion-compensated prediction block **150**. The combiner **112** combines each of the locally decoded blocks of prediction error values with a corresponding block of predicted pixel values to produce reconstructed image blocks and stores them in frame store **120**.

As subsequent macroblocks of the video signal are received from the video source and undergo the previously described encoding and decoding steps in blocks **104**, **106**, **108**, **110**, **112**, a decoded version of the frame is built up in frame store **120**. When the last macroblock of the frame has been processed, the frame store **120** contains a completely decoded frame, available for use as a motion prediction ref-

reference frame in encoding a subsequently received video frame in INTER-coded format.

The details of the motion-compensated prediction performed by video encoder **100** will now be considered.

Any frame encoded in INTER-coded format requires a reference frame for motion-compensated prediction. This means, necessarily, that when encoding a video sequence, the first frame to be encoded, whether it is the first frame in the sequence, or some other frame, must be encoded in INTRA-coded format. This, in turn, means that when the video encoder **100** is switched into INTER-coding mode by control manager **160**, a complete reference frame, formed by locally decoding a previously encoded frame, is already available in the frame store **120** of the encoder. In general, the reference frame is formed by locally decoding either an INTRA-coded frame or an INTER-coded frame.

In the following description it will be assumed that the encoder performs motion compensated prediction on a macroblock basis, i.e. a macroblock is the smallest element of a video frame that can be associated with motion information. It will further be assumed that a prediction for a given macroblock is formed by identifying a region of 16×16 values in the luminance component of the reference frame that shows best correspondence with the 16×16 luminance values of the macroblock in question. Motion-compensated prediction in a video coding system where motion information may be associated with elements smaller than a macroblock will be considered later in the text.

The first step in forming a prediction for a macroblock of the current frame is performed by motion estimation block **130**. The motion estimation block **130** receives the blocks of luminance and chrominance values which make up the current macroblock of the frame to be coded via line **128**. It then performs a block matching operation in order to identify a region in the reference frame that corresponds best with the current macroblock. In order to perform the block matching operation, motion estimation block **130** accesses reference frame data stored in frame store **120** via line **127**. More specifically, motion estimation block **130** performs block-matching by calculating difference values (e.g. sums of absolute differences) representing the difference in pixel values between the macroblock under examination and candidate best-matching regions of pixels from a reference frame stored in the frame store **120**. A difference value is produced for candidate regions at all possible offsets within a predefined search region of the reference frame and motion estimation block **130** determines the smallest calculated difference value. The candidate region that yields the smallest difference value is selected as the best-matching region. The offset between the current macroblock and the best-matching region identified in the reference frame defines a "motion vector" for the macroblock in question. The motion vector typically comprises a pair of numbers, one describing the horizontal (Δx) between the current macroblock and the best-matching region of the reference frame, the other representing the vertical displacement (Δy).

Once the motion estimation block **130** has produced a motion vector for the macroblock, it outputs the motion vector to the motion field coding block **140**. The motion field coding block **140** approximates the motion vector received from motion estimation block **130** using a motion model comprising a set of basis functions and motion coefficients. More specifically, the motion field coding block **140** represents the motion vector as a set of motion coefficient values which, when multiplied by the basis functions, form an approximation of the motion vector. Typically, a translational

motion model having only two motion coefficients and basis functions is used, but motion models of greater complexity may also be used.

The motion coefficients are passed from motion field coding block **140** to motion compensated prediction block **150**. Motion compensated prediction block **150** also receives the best-matching region of pixel values identified by motion estimation block **130** from frame store **120**. Using the approximate representation of the motion vector generated by motion field coding block **140** and the pixel values of the best-matching region of pixels from the reference frame, motion compensated prediction block **150** generates an array of predicted pixel values for each block of the current macroblock. Each block of predicted pixel values is passed to combiner **116** where the predicted pixel values are subtracted from the actual (input) pixel values in the corresponding block of the current macroblock. In this way a set of prediction error blocks for the macroblock is obtained.

Operation of the video decoder **200**, shown in FIG. **2** will now be described. The decoder **200** comprises a video multiplex decoder **270**, which receives an encoded video bit-stream **135** from the encoder **100** and demultiplexes it into its constituent parts, an inverse quantizer **210**, an inverse DCT transformer **220**, a motion compensated prediction block **240**, a frame store **250**, a combiner **230**, a control manager **260**, and an output **280**.

The control manager **260** controls the operation of the decoder **200** in response to whether an INTRA- or an INTER-coded frame is being decoded. An INTRA/INTER trigger control signal, which causes the decoder to switch between decoding modes is derived, for example, from picture type information associated with each compressed video frame received from the encoder. The INTRA/INTER trigger control signal is extracted from the encoded video bit-stream by the video multiplex decoder **270** and is passed to control manager **260** via control line **215**.

Decoding of an INTRA-coded frame is performed on a macroblock-by-macroblock basis, each macroblock being decoded substantially as soon as encoded information relating to it is received in the video bit-stream **135**. The video multiplex decoder **270** separates the encoded information for the blocks of the macroblock from possible control information relating to the macroblock in question. The encoded information for each block of an INTRA-coded macroblock comprises variable length codewords representing the entropy coded level and run values for the non-zero quantized DCT coefficients of the block. The video multiplex decoder **270** decodes the variable length codewords using a variable length decoding method corresponding to the encoding method used in the encoder **100** and thereby recovers the level and run values. It then reconstructs the array of quantized transform coefficient values for each block of the macroblock and passes them to inverse quantizer **210**. Any control information relating to the macroblock is also decoded in the video multiplex decoder **270** using an appropriate decoding method and is passed to control manager **260**. In particular, information relating to the level of quantization applied to the transform coefficients is extracted from the encoded bit-stream by video multiplex decoder **270** and provided to control manager **260** via control line **217**. The control manager, in turn, conveys this information to inverse quantizer **210** via control line **218**. Inverse quantizer **210** inverse quantizes the quantized DCT coefficients for each block of the macroblock according to the control information and provides the now inverse quantized DCT coefficients to inverse DCT transformer **220**.

Inverse DCT transformer **220** performs an inverse DCT transform on the inverse quantized DCT coefficients for each

block of the macroblock to form a decoded block of image information comprising reconstructed pixel values. The reconstructed pixel values for each block of the macroblock are passed via combiner 230 to the video output 280 of the decoder where, for example, they can be provided to a display device (not shown). The reconstructed pixel values for each block are also stored in frame store 250. Because motion-compensated prediction is not used in the encoding/decoding of INTRA coded macroblocks control manager 260 controls combiner 230 to pass each block of pixel values as such to the video output 280 and frame store 250. As subsequent macroblocks of the INTRA-coded frame are decoded and stored, a decoded frame is progressively assembled in the frame store 250 and thus becomes available for use as a reference frame for motion compensated prediction in connection with the decoding of subsequently received INTER-coded frames.

INTER-coded frames are also decoded macroblock by macroblock, each INTER-coded macroblock being decoded substantially as soon as encoded information relating to it is received in the bit-stream 135. The video multiplex decoder 270 separates the encoded prediction error information for each block of an INTER-coded macroblock from encoded motion vector information and possible control information relating to the macroblock in question. As explained in the foregoing, the encoded prediction error information for each block of the macroblock comprises variable length code-words representing the entropy coded level and run values for the non-zero quantized transform coefficients of the prediction error block in question. The video multiplex decoder 270 decodes the variable length codewords using a variable length decoding method corresponding to the encoding method used in the encoder 100 and thereby recovers the level and run values. It then reconstructs an array of quantized transform coefficient values for each prediction error block and passes them to inverse quantizer 210. Control information relating to the INTER-coded macroblock is also decoded in the video multiplex decoder 270 using an appropriate decoding method and is passed to control manager 260. Information relating to the level of quantization applied to the transform coefficients of the prediction error blocks is extracted from the encoded bit-stream and provided to control manager 260 via control line 217. The control manager, in turn, conveys this information to inverse quantizer 210 via control line 218. Inverse quantizer 210 inverse quantizes the quantized DCT coefficients representing the prediction error information for each block of the macroblock according to the control information and provides the now inverse quantized DCT coefficients to inverse DCT transformer 220. The inverse quantized DCT coefficients representing the prediction error information for each block are then inverse transformed in the inverse DCT transformer 220 to yield an array of reconstructed prediction error values for each block of the macroblock.

The encoded motion vector information associated with the macroblock is extracted from the encoded video bit-stream 135 by video multiplex decoder 270 and is decoded. The decoded motion vector information thus obtained is passed via control line 225 to motion compensated prediction block 240, which reconstructs a motion vector for the macroblock using the same motion model as that used to encode the INTER-coded macroblock in encoder 100. The reconstructed motion vector approximates the motion vector originally determined by motion estimation block 130 of the encoder. The motion compensated prediction block 240 of the decoder uses the reconstructed motion vector to identify the location of a region of reconstructed pixels in a prediction reference frame stored in frame store 250. The reference frame may be, for example, a previously decoded INTRA-

coded frame, or a previously decoded INTER-coded frame. In either case, the region of pixels indicated by the reconstructed motion vector is used to form a prediction for the macroblock in question. More specifically, the motion compensated prediction block 240 forms an array of pixel values for each block of the macroblock by copying corresponding pixel values from the region of pixels identified by the motion vector. The prediction, that is the blocks of pixel values derived from the reference frame, are passed from motion compensated prediction block 240 to combiner 230 where they are combined with the decoded prediction error information. In practice, the pixel values of each predicted block are added to corresponding reconstructed prediction error values output by inverse DCT transformer 220. In this way an array of reconstructed pixel values for each block of the macroblock is obtained. The reconstructed pixel values are passed to the video output 280 of the decoder and are also stored in frame store 250. As subsequent macroblocks of the INTER-coded frame are decoded and stored, a decoded frame is progressively assembled in the frame store 250 and thus becomes available for use as a reference frame for motion-compensated prediction of other INTER-coded frames.

As explained above, in a typical video coding system, motion compensated prediction is performed on a macroblock basis, such that a macroblock is the smallest element of a video frame that can be associated with motion information. However, the video coding recommendation currently being developed by the Joint Video Team (JVT) of ISO/IEC MPEG (Motion Pictures Expert Group) and ITU-T VCEG (Video Coding Experts Group), allows motion information to be associated with elements smaller than a macroblock. In the following description, and throughout the remainder of the text, reference will be made to the version of this video coding standard described in the document by T. Weigand: "Joint Model Number 1", Doc. JVT-A003, Joint Video Team of ISO/IEC MPEG and ITU-T VCEG, January 2002, said document being included herein in its entirety. For simplicity, this version of the recommendation will be referred to as "JM1 of the JVT codec".

According to JM1 of the JVT codec, video pictures are divided into macroblocks of 16x16 pixels and are coded on a macroblock-by-macroblock basis. The coding performed follows the basic principles described above in connection with the generic video encoder and decoder of FIGS. 1 and 2. However, according to JM1, motion compensated prediction of INTER coded macroblocks is performed in manner that differs from that previously described. More specifically, each of the macroblocks is assigned a "coding mode" depending on the characteristics of the macroblock and the motion in the video sequence. Seven of the coding modes are based on dividing a macroblock to be INTER coded into a number of sub-blocks, each comprising NxM pixels, and associating motion information with each of the NxM sub-blocks, not just with the macroblock as a whole. Each of the possible schemes for dividing a macroblock into NxM sub-blocks, provided by JM1 of the JVT video codec, is illustrated in FIG. 4 of the accompanying drawings. As can be seen from the figure, the possible divisions are: 16x16, 8x16, 16x8, 8x8, 4x8, 8x4 and 4x4. Thus, if the coding mode assigned to a particular macroblock is, for example, the 16x8 mode, the macroblock is divided into two sub-blocks of size 16x8 pixels each and both sub-blocks is provided with its own motion information. In addition, an eighth coding mode, known as SKIP (or skip) mode, is provided. If this mode is assigned to a macroblock, this indicates that the macroblock is to be copied from the reference video frame without using motion compensated prediction.

13

guide the building of new reference frames for global motion compensation. More specifically, the global motion compensation scheme employed in the H.263 video coding standard uses a resampling process to generate a warped version of the reference frame for use in motion-compensated prediction of the current picture. This warped version of the reference frame may include alterations in the shape, size, and location with respect to the current picture. The resampling process is defined in terms of a mapping between the four corners of the current picture and the corresponding four corners of the reference frame. Assuming that the luminance component of the current picture has a horizontal size H and vertical size V, the mapping is performed by defining four conceptual motion vectors v^{OO} , v^{HO} , v^{OV} , and v^{HV} , each conceptual motion vector describing how to move one of the four corners of the current picture in such a way as to map it onto the corresponding corner of the reference frame. This operation is illustrated in FIG. 5. Motion compensated prediction for a macroblock of the current picture is then performed using block-matching with respect to the warped reference frame. This complicates the block matching process, as the value of each pixel of the warped reference frame used in the block matching process must be generated by mapping pixel values in the original (non-warped) reference frame into the co-ordinates of the warped reference frame. This is done using bilinear interpolation, which is a computationally intensive operation. The reader is referred to Annex P of the H.263 video coding standard for further details of the bilinear interpolation process used to generate the pixel values of the warped reference frame.

Global motion vector coding, as described in document VCEG-O20, referred to above, is a simplified version of global motion compensation. The reference frame is used as it is, but additional information is transmitted to describe the global motion and additional macroblock modes are used to indicate when global motion vectors are used. This approach is less complex than the global motion compensation technique just described, but there is additional encoder complexity associated with it. Namely, the encoder must perform extra motion estimation operations to find the global motion parameters and it also needs to evaluate more macroblock modes to find the optimal one. Moreover, the amount of extra global motion information that needs to be transmitted becomes large for small resolution video.

In view of the preceding discussion, it should be appreciated that there exists a significant unresolved technical problem relating to the coding of a digital video sequence in the presence of global motion, such as translation, panning or zooming of the camera. In particular, each of the three previously described prior art video coding solutions has some form of technical shortcoming. JM1 of the JVT codec, for example, has no special provision for taking account of global motion in video sequences. Therefore, when such motion is present it causes the video encoder to select macroblock coding modes that explicitly model the motion. This leads to a significant degradation in coding efficiency, as the global motion component is encoded in every INTER coded macroblock (or sub-block). The technique of global motion compensation (as provided by Annex P of the H.263 video coding standard) takes global motion into account by warping reference frames used in motion compensated prediction and therefore provides improved coding efficiency compared with a system in which no special measures are taken to code global motion. However, the warping process is computationally complex and additional information must be transmitted in the encoded video bit-stream to enable correct decoding of the video sequence. Although the related technique of global

14

motion vector coding is computationally less demanding than global motion compensation, it does involve a certain increase in encoder complexity and additional information must be still transmitted in the video bit-stream to enable correct decoding of the video data.

It is therefore one purpose of the present invention to combine the simplicity of local motion compensation with the coding efficiency of global motion compensation to yield a video coding system with significantly improved compression performance and a negligible increase in complexity.

SUMMARY OF THE INVENTION

In order to overcome, or at least mitigate to a large extent the problems associated with the coding of global motion in prior art video coding systems, the present invention is based on a redefinition of the skip mode concept used in JM1 of the JVT codec. The method according to the invention not only provides an improvement in coding efficiency in the presence of global motion (i.e. motion affecting the entire area of video frame), but also enables regional motion to be represented in an efficient manner.

According to the invention, the skip mode concept is redefined in such a way that a macroblock assigned to skip mode is either associated with a zero (non-active) motion vector, in which case it is treated in the same way as a conventional skip mode macroblock and copied directly from the reference frame, or it is associated with a non-zero (active) motion vector. The decision as to whether a macroblock should be associated with a zero or non-zero motion vector is made by analysing the motion of other macroblocks or sub-blocks in a region surrounding the macroblock to be coded. If it is found that the surrounding region exhibits a certain type of motion, a non-zero motion vector representative of that motion is generated and associated with the current macroblock. In particular, the continuity, velocity or deviation of motion in the surrounding macroblocks or sub-blocks can be analyzed. For example, if the motion in the surrounding region exhibits a certain level of continuity, a certain common velocity, or a particular form of divergence, a motion vector representative of that motion can be assigned to the current macroblock to be coded. On the other hand, if the region surrounding the current macroblock does not exhibit such continuity, common velocity or divergence and has an insignificant level of motion, the macroblock to be coded is assigned a zero motion vector, causing it to be copied directly from the reference frame, just as if it were a conventional SKIP mode macroblock. In this way, according to the invention, SKIP mode macroblocks can adapt to the motion in the region surrounding them, enabling global or regional motion to be taken account of in an efficient manner.

In an advantageous embodiment of the invention, the surrounding macroblocks or sub-blocks whose motion is analysed are previously encoded macroblocks neighboring the macroblock to be coded. This ensures that motion information relating to the region the surrounding a macroblock is available in the encoder (decoder) when a current macroblock is being encoded (decoded) and can be used directly to determine the motion vector to be assigned to the current macroblock. This approach enables the motion analysis of the surrounding region performed in the encoder to be duplicated exactly in the decoder. This, in turn, means that according to the invention, no additional information must be sent to the decoder in order to model global or regional motion.

As will become apparent from the detailed description of the invention presented below, redefinition of the skip mode concept as proposed by the present invention has significant

15

technical advantages compared with the previously described prior art video coding methods. In particular, the method according to the invention enables global and regional motion within a video sequence to be taken account of in an efficient manner without the need for complex warping of the reference frame or any other computationally demanding operations. Furthermore, in contrast to both the global motion compensation and global motion vector coding methods previously described, no additional information must be transmitted in the video bit-stream to enable correct decoding of the video data. Additionally, a minimal amount of modification is required to incorporate the method according to the invention into existing video coding systems that employ the concept of skip mode macroblocks.

These and other features, aspects, and advantages of embodiments of the present invention will become apparent with reference to the following detailed description in conjunction with the accompanying drawings. It is to be understood, however, that the drawings are designed solely for the purposes of illustration and not as a definition of the limits of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a schematic block diagram of a generic video encoder according to prior art.

FIG. 2 is a schematic block diagram of a generic video decoder according to prior art and corresponding to the encoder shown in FIG. 1.

FIG. 3 illustrates the formation of a macroblock according to prior art.

FIG. 4 shows the 7 possible divisions of macroblocks into blocks according to JM1 of the JVT video codec.

FIG. 5 illustrates the generation of conceptual motion vectors for mapping the corners of a current picture to those of a reference picture in the global motion compensation scheme according to H.263 Annex P.

FIG. 6 is a schematic block diagram of a video encoder according to an embodiment of the invention.

FIG. 7 is a schematic block diagram of a video decoder according to an embodiment of the invention and corresponding to the encoder shown in FIG. 6.

FIG. 8 illustrates encoding and decoding blocks for skip mode macroblocks in an encoder or decoder according to an embodiment of the invention.

FIG. 9 shows an example of macroblock partitioning, motion in macroblocks surrounding a macroblock to be coded or decoded, and the generated skip mode motion vector for the macroblock (the darkened macroblock in the figure) according to an embodiment of the invention.

FIG. 10 is a schematic block diagram of a multimedia communications terminal in which the method according to the invention may be implemented.

BEST MODE FOR CARRYING OUT THE INVENTION

Exemplary embodiments of the invention will now be described in detail with particular reference to FIGS. 6 to 10.

According to the invention, skip (or SKIP) mode macroblocks in a video coding system adapt to the motion of surrounding image segments. If active motion is detected around a macroblock to be coded/decoded, motion parameters conforming to the motion are generated and the macroblock is motion compensated. In this way, no additional information needs to be transmitted from the encoder to the decoder.

16

FIG. 6 is a schematic block diagram of a video encoder 600 implemented according to an embodiment of the invention. When encoding frames of a digital video sequence, encoder 600 operates in a manner similar to that previously described in connection with the prior art video encoder of FIG. 1 to generate INTRA-coded and INTER-coded compressed video frames. The structure of the video encoder shown in FIG. 6 is substantially identical to that of the prior art video encoder shown in FIG. 1, with appropriate modifications to the motion estimation part necessary to implement the video encoding method according to the invention. All parts of the video encoder which implement functions and operate in a manner identical to the previously described prior art video encoder are identified with identical reference numbers.

As the present invention relates to the encoding of video frames in INTER-coded format and more particularly to details of the motion-compensated prediction performed as part of the INTER coding process, description of encoder 600 in INTRA-coding mode will be omitted and the following sections will concentrate on the operations performed by the encoder in INTER-coding mode.

In INTER-coding mode, the video encoder's control manager 160 operates switch 102 to receive its input from line 117, which comprises the output of combiner 116. The combiner 116 receives the video input signal macroblock by macroblock from input 101. As combiner 116 receives the blocks of luminance and chrominance values which make up the macroblock, it forms corresponding blocks of prediction error information, representing the difference between the block in question and its prediction, produced in motion compensated prediction block 650.

The prediction error information for each block of the macroblock is passed to DCT transformation block 104, which performs a two-dimensional discrete cosine transform on each block of prediction error values to produce a two-dimensional array of DCT transform coefficients for each block. These are passed to quantizer 106 where they are quantized using a quantization parameter QP. Selection of the quantization parameter QP is controlled by the control manager 160 via control line 115.

The quantized DCT coefficients representing the prediction error information for each block of the macroblock are then passed from quantizer 106 to video multiplex coder 170, via line 125. The video multiplex coder 170 orders the transform coefficients for each prediction error block using a zig-zag scanning procedure, represents each non-zero valued quantized coefficient as a run-level pair and compresses the run-level pairs using entropy coding. Video multiplex coder 170 also receives motion vector information from motion field coding block 640 via line 126 and control information from control manager 160. It entropy codes the motion vector information and control information and forms a single bit-stream of coded image information, 135 comprising the entropy coded motion vector, prediction error and control information.

The quantized DCT coefficients representing the prediction error information for each block of the macroblock are also passed from quantizer 106 to inverse quantizer 108. Here they are inverse quantized and the resulting blocks of inverse quantized DCT coefficients are applied to inverse DCT transform block 110, where they undergo inverse DCT transformation to produce locally decoded blocks of prediction error values. The locally decoded blocks of prediction error values are then input to combiner 112. In INTER-coding mode, switch 114 is set so that the combiner 112 also receives predicted pixel values for each block of the macroblock, generated by motion-compensated prediction block 650. The

combiner **112** combines each of the locally decoded blocks of prediction error values with a corresponding block of predicted pixel values to produce reconstructed image blocks and stores them in frame store **120**.

As subsequent macroblocks of the video signal are received from the video source and undergo the previously described encoding and decoding steps in blocks **104**, **106**, **108**, **110**, **112**, a decoded version of the frame is built up in frame store **120**. When the last macroblock of the frame has been processed, the frame store **120** contains a completely decoded frame, available for use as a motion prediction reference frame in encoding a subsequently received video frame in INTER-coded format.

The details of the motion-compensated prediction performed by video encoder **600** will now be described in detail.

Encoder **600** performs motion-compensated prediction in a manner similar to the previously described JVT codec. In other words, it is adapted to assign a coding mode to each INTER-coded macroblock depending on the characteristics of the macroblock and the motion in the video sequence being coded. When examining which coding mode to assign to particular macroblock, motion estimation block **630** performs a motion estimation operation for each coding mode in turn. Motion estimation block **630** receives the blocks of luminance and chrominance values which make up the macroblock to be coded for use in motion estimation via line **128** (see FIG. **6**). It then selects each of the possible coding modes one after the other, in turn, and performs motion estimation in order to identify a best match for the macroblock in the reference frame, on the basis of the selected coding mode and the pixel values of the macroblock to be coded. (The best match will comprise one or more best-matching regions of pixel values, depending on the coding mode). Each best-match is associated with an overall cost value, for example, a linear combination of the sum of absolute differences between the pixel values in the macroblock under examination and the best matching region in the reference frame, and an estimated number of bits required to code the mode and represent motion vectors. Once a best match has been obtained for each coding mode, motion estimation block **630** selects that coding mode which yields the smallest overall cost value as the coding mode for the current macroblock.

According to the invention, the coding modes used by encoder **600** correspond to those provided by JM1 of the JVT codec (shown in Table 3), with the exception that the SKIP mode is redefined to allow representation of global and regional motion. More specifically, the SKIP mode is modified in such a way that a zero (non-active) motion vector or a non-zero (active) motion vector is associated with each skip mode macroblock, depending on the characteristics of the motion in image segments surrounding the macroblock in question. In the following this type of motion vector will be referred to as a "skip mode motion vector".

When examining skip mode as part of the previously described motion estimation process performed in turn for each coding mode, the encoder first determines whether a zero or a non-zero skip mode motion vector should be used. To do this, the encoder is arranged to analyze the motion of image segments (e.g. macroblocks and/or sub-blocks) surrounding the macroblock to be coded. If it determines that the surrounding region exhibits a certain type of motion, for example it has characteristics indicative of global or regional motion, it generates a non-zero valued skip mode motion vector representative of the motion. On the other hand, if the encoder determines that the-region surrounding the current macroblock does not exhibit global or regional motion, but instead has an insignificant level of motion, it generates a zero

valued skip mode motion vector. In other words, if the encoder determines that the motion in the region surrounding the current macroblock has a global characteristic, skip mode coding is adapted to take account of this (by generating an associated non-zero valued skip mode motion vector representative of the motion). Alternatively if no such motion is present, a zero valued motion vector is generated causing the skip mode as modified by the invention to operate in a conventional manner i.e. a zero valued skip mode motion vector causes a macroblock to be copied directly from the reference frame.

Having performed motion estimation operations for each of the available coding modes, including skip mode as modified according to the invention, encoder **600** determines which coding mode yields the smallest overall cost value and selects that mode as the coding mode for the macroblock in question. An indication of the finally selected coding mode, for example a variable length codeword selected from the set of codewords presented in Table 3, is associated with the macroblock and included in the video bit-stream **635**. This enables a corresponding decoder to identify the coding mode for the macroblock and correctly reconstruct the macroblock using the correct form of motion-compensated prediction.

The analysis of motion in a region surrounding a macroblock to be coded to determine whether a zero valued or non-zero valued skip mode motion vector should be used will now be considered in further detail with reference to FIG. **8** of the accompanying drawings. FIG. **8** illustrates the functional elements of the motion estimation block **630** associated with generating skip mode motion vectors. These include motion information memory **801**, surrounding motion analysis block **802**, active motion parameter generation block **803** and zero motion parameter generation block **804**.

The decision whether to generate a zero valued skip mode motion vector or a non-zero valued skip mode motion vector is made by surrounding motion analysis block **802**. The decision is made by analysing and classifying the motion of macroblocks or sub-blocks in a predefined region surrounding the macroblock to be coded using a predetermined analysis scheme. In order to perform the analysis, surrounding motion analysis block **802** retrieves motion information relating to the macroblocks and/or sub-blocks in the surrounding region from motion information memory **801**. Depending on the specific details of the implementation, surrounding motion analysis block may be arranged to analyze the continuity, velocity or deviation of motion in the surrounding macroblocks or sub-blocks. For example, if the motion in the surrounding region exhibits a certain level of continuity, a certain common velocity (as depicted in FIG. **9**, for example), or a particular form of divergence, this may suggest that some form of global or regional motion is present. As a consequence surrounding motion analysis block concludes that "active motion" is present in the surrounding region and a non-zero valued skip mode motion vector should be used. On the other hand, if the region surrounding the current macroblock does not exhibit such continuity, common velocity or divergence and has a generally insignificant level of motion, the surrounding motion analysis block concludes that "non-active motion" is present in the surrounding region and consequently a zero valued skip mode motion vector should be used.

As shown in FIG. **8**, if the surrounding motion analysis block determines that "active motion" is present in the surrounding region, it sends an indication to that effect to active motion parameter generation block **803**, which forms a non-zero valued skip mode motion vector representative of the motion in the surrounding region. To do this active motion

parameter generation block **803** retrieves motion information relating to the surrounding macroblocks and/or sub-blocks from motion information memory **801**. Alternatively, this information may be passed to the active motion parameter generation block by surrounding motion analysis block **802**. If surrounding motion analysis block determines that “non-active motion” is present in the surrounding region, it sends a corresponding indication to zero motion parameter generation block **804**, which forms a zero valued skip mode motion vector.

In a particularly advantageous embodiment of the invention, the surrounding region of macroblocks or sub-blocks analyzed by the surrounding motion analysis block comprises previously encoded macroblocks neighboring the macroblock to be coded (FIG. 9). In this case, the analysis and classification of motion in the surrounding region performed in the encoder can be duplicated exactly in the decoder. This, in turn, means that according to the invention, no additional information must be sent to the decoder in order to model global or regional motion.

In an alternative embodiment of the invention the coding modes of already coded macroblocks are taken into account when deciding whether to use a zero valued or non-zero valued skip mode motion vector. For example, if the surrounding motion analysis block determines that there is one or more stationary neighboring macroblock, a zero valued skip mode motion vector is used.

In a first preferred embodiment of the invention surrounding motion analysis block **802** classifies the motion in the region surrounding the macroblock according to the following three step procedure. Firstly, surrounding motion analysis block retrieves motion information for the macroblocks or sub-blocks surrounding the macroblock to be coded (i.e. previously encoded macroblocks neighboring the macroblock to be coded, as shown in FIG. 9) and generates a median motion vector prediction for the macroblock. The median motion vector prediction is formed, for example, in a manner analogous to that used in motion vector prediction according to JM1 of the JVT codec (see T. Weigand: “Joint Model Number 1”, Doc. JVT-A003, Joint Video Team of ISO/IEC MPEG and ITU-T VCEG, January 2002). Next surrounding motion analysis block determines if any of the resulting motion vector components has an absolute value larger than a certain threshold value (for example half a pixel). If this condition is fulfilled, the motion is classified as “active motion”, otherwise it is classified as “non-active motion”. Finally, depending on the classification result, surrounding motion analysis block **802**, sends an indication to either the active motion parameter generation block **803** or the zero motion parameter generation block **804** to in order to generate the appropriate skip mode motion parameters.

Implementation of the surrounding motion analysis block according to the first preferred embodiment of the invention is particularly advantageous for two reasons. Firstly, in a typical video codec, such as the JVT codec, a median predictor is used to predict motion vectors of square image blocks. According to the first preferred embodiment, this same predictor is used in the surrounding motion analysis block and active motion parameter generation block to analyze motion in the region surrounding a macroblock to be coded and to generate motion parameters for SKIP mode macroblocks. In this way the invention can be implemented with minimal effect on the total implementation complexity of the video codec. Secondly, because the surrounding motion analysis block **802** classifies the motion in the surrounding region by generating and analyzing a median motion vector, active motion parameter generation block **803** can simply pass the

median motion parameters, already generated in the surrounding motion analysis block, to the motion compensation block. This also minimizes the implementation complexity, since there is no need to generate additional motion parameters.

In a second preferred embodiment of the invention the surrounding motion analysis block analyses the motion in vicinity of the macroblock to be coded and classifies it as either “active motion” or “non-active motion”. In the case of “active motion” the active motion parameter generation block is activated and in the case of “non-active motion” the zero motion parameter generation block is activated. In this embodiment the classification to the “non-active motion” category takes place if either or both of the two conditions below are true, otherwise the motion is classified as “active motion”:

Condition 1: The macroblock immediately above or the macroblock immediately to the left of the macroblock under consideration is not available (that is, is out of the picture or belongs to a different slice).

Condition 2: The macroblock or block immediately above, or the macroblock or block immediately to the left that are used in motion vector prediction for the 16×16 INTER mode has a zero motion vector and uses the latest picture as reference in motion compensation.

Operation of a video decoder **700** according to an embodiment of the invention will now be described with reference to FIG. 7. The structure of the video decoder illustrated in FIG. 7 is substantially identical to that of the prior art video decoder shown in FIG. 2, with appropriate modifications to those parts of the decoder that perform motion estimation operations. All parts of the video decoder which implement functions and operate in a manner identical to the previously described prior art video decoder are identified with identical reference numbers. It is further assumed that the video decoder of FIG. 7 corresponds to the encoder described in connection with FIG. 6 and is therefore capable of receiving and decoding the bit-stream **635** transmitted by encoder **600**. Furthermore, as the present invention affects the decoding of video frames in INTER-coded format, description of the operations performed by decoder **700** in connection with the decoding of INTRA-coded frames will be omitted.

INTER-coded frames are decoded macroblock by macroblock, each INTER-coded macroblock being decoded substantially as soon as encoded information relating to it is received in the bit-stream **635**. Depending on the coding mode, the compressed video data included in the bit-stream for an INTER-coded macroblock may comprise a combination of VLC encoded prediction error information for each block, motion vector information for the macroblock (or sub-blocks) and encoded control information including an indication of the coding mode used to encode the macroblock in question. If a macroblock is encoded in skip mode, no prediction error or motion vector information relating to the macroblock is included in the bit-stream.

Video multiplex decoder **270** receives the video bit-stream **635** and separates control information, including an indication of the coding mode of the macroblock from any encoded prediction error and/or motion vector information that may be present.

As explained earlier, prediction error information is encoded as variable length codewords representative of entropy coded level and run values. If prediction error information is provided for the current macroblock, the video multiplex decoder **270** recovers the level and run values by decoding the variable length codewords using a variable length decoding method corresponding to the encoding

21

method used in encoder **600**. It then reconstructs an array of quantized DCT transform coefficient values for each prediction error block and passes them to inverse quantizer **210** where they are inverse quantized. The inverse quantized DCT coefficients are then inverse transformed in the inverse DCT transformer **220** to yield an array of reconstructed prediction error values for each block of the macroblock.

Both the coding mode indication and encoded motion vector information (if any) associated with the macroblock are decoded in the video multiplex decoder and are passed via control line **225** to motion compensated prediction block **740**. Motion compensated prediction block **740** uses the coding mode indication and motion vector information (if any) to form a prediction for the macroblock in question. More specifically, the motion compensated prediction block **740** forms an array of pixel values for each block of the macroblock by copying corresponding pixel values from a region (or regions) of pixels in a reference frame. The prediction, that is the blocks of pixel values derived from the reference frame, are passed from motion compensated prediction block **740** to combiner **230** where they are combined with the decoded prediction error information (if any). In this way an array of reconstructed pixel values for each block of the macroblock is obtained.

The reconstructed pixel values are passed to the video output **280** of the decoder and are also stored in frame store **250**. Consequently, as subsequent macroblocks of the INTER-coded frame are decoded and stored, a decoded frame is progressively assembled in the frame store **250** and thus becomes available for use as a reference frame for motion-compensated prediction of other INTER-coded frames.

According to the invention, the motion compensated prediction block **740** of decoder **700** comprises a motion information memory block **801**, a surrounding motion analysis block **802**, an active motion parameter generation block **803** and a zero motion parameter generation block **804** analogous to those provided in encoder **600**. These functional blocks are used to determine whether a macroblock encoded in skip mode should be associated with a zero valued or a non-zero valued skip mode motion vector. More specifically, when it is determined that a macroblock to be decoded was encoded in skip mode, surrounding motion analysis block **802** analyses and classifies the motion of previously decoded macroblocks and/or sub-blocks in a predefined region surrounding the macroblock to be decoded in a manner exactly corresponding to that used in encoder **600**. As a result of the analysis, the macroblock in question is either associated with a non-zero valued skip mode motion vector or a zero valued skip mode motion vector. This motion vector is then used to form a prediction for the macroblock. If the macroblock is associated with a zero valued skip mode motion vector, it is reconstructed by simply copying pixel values from a corresponding location in the reference frame. If, on the other hand, it is associated with a non-zero valued motion vector, a region of pixel values indicated by the non-zero motion vector is used to generate the pixel values for the macroblock.

It should be appreciated that by modifying the skip mode concept in the manner proposed by the invention and performing surrounding motion analysis in the decoder, it is possible to take account of global or region motion in a video sequence without requiring explicit information about such motion to be provided in video bit-stream.

A terminal device comprising video encoding and decoding equipment which may be adapted to operate in accordance with the present invention will now be described. FIG. **10** of the accompanying drawings illustrates a multimedia terminal **80** implemented according to ITU-T recommenda-

22

tion H.324. The terminal can be regarded as a multimedia transceiver device. It includes elements that capture, encode and multiplex multimedia data streams for transmission via a communications network, as well as elements that receive, de-multiplex, decode and display received multimedia content. ITU-T recommendation H.324 defines the overall operation of the terminal and refers to other recommendations that govern the operation of its various constituent parts. This kind of multimedia terminal can be used in real-time applications such as conversational videotelephony, or non real-time applications such as the retrieval and/or streaming of video clips, for example from a multimedia content server in the Internet.

In the context of the present invention, it should be appreciated that the H.324 terminal shown in FIG. **10** is only one of a number of alternative multimedia terminal implementations suited to application of the inventive method. It should also be noted that a number of alternatives exist relating to the location and implementation of the terminal equipment. As illustrated in FIG. **10**, the multimedia terminal may be located in communications equipment connected to a fixed line telephone network such as an analogue PSTN (Public Switched Telephone Network). In this case the multimedia terminal is equipped with a modem **91**, compliant with ITU-T recommendations V.8, V.34 and optionally V.8bis. Alternatively, the multimedia terminal may be connected to an external modem. The modem enables conversion of the multiplexed digital data and control signals produced by the multimedia terminal into an analogue form suitable for transmission over the PSTN. It further enables the multimedia terminal to receive data and control signals in analogue form from the PSTN and to convert them into a digital data stream that can be demultiplexed and processed in an appropriate manner by the terminal.

An H.324 multimedia terminal may also be implemented in such a way that it can be connected directly to a digital fixed line network, such as an ISDN (Integrated Services Digital Network). In this case the modem **91** is replaced with an ISDN user-network interface. In FIG. **10**, this ISDN user-network interface is represented by alternative block **92**.

H.324 multimedia terminals may also be adapted for use in mobile communication applications. If used with a wireless communication link, the modem **91** can be replaced with any appropriate wireless interface, as represented by alternative block **93** in FIG. **10**. For example, an H.324/M multimedia terminal can include a radio transceiver enabling connection to the current 2nd generation GSM mobile telephone network, or the proposed 3rd generation UMTS (Universal Mobile Telephone System).

It should be noted that in multimedia terminals designed for two-way communication, that is for transmission and reception of video data, it is advantageous to provide both a video encoder and video decoder implemented according to the present invention. Such an encoder and decoder pair is often implemented as a single combined functional unit, referred to as a "codec".

A typical H.324 multimedia terminal will now be described in further detail with reference to FIG. **10**.

The multimedia terminal **80** includes a variety of elements referred to as "terminal equipment". This includes video, audio and telematic devices, denoted generically by reference numbers **81**, **82** and **83**, respectively. The video equipment **81** may include, for example, a video camera for capturing video images, a monitor for displaying received video content and optional video processing equipment. The audio equipment **82** typically includes a microphone, for example for capturing spoken messages, and a loudspeaker for reproducing received

audio content. The audio equipment may also include additional audio processing units. The telematic equipment **83**, may include a data terminal, keyboard, electronic whiteboard or a still image transceiver, such as a fax unit.

The video equipment **81** is coupled to a video codec **85**. The video codec **85** comprises a video encoder and a corresponding video decoder, both implemented according to the invention. Such an encoder and a decoder will be described in the following. The video codec **85** is responsible for encoding captured video data in an appropriate form for further transmission over a communications link and decoding compressed video content received from the communications network. In the example illustrated in FIG. **10**, the video codec is implemented according to JM1 of the JVT codec, with appropriate modifications to implement the modified SKIP mode concept according to the invention in both the encoder and the decoder of the video codec.

The terminal's audio equipment is coupled to an audio codec, denoted in FIG. **10** by reference number **86**. Like the video codec, the audio codec comprises an encoder/decoder pair. It converts audio data captured by the terminal's audio equipment into a form suitable for transmission over the communications link and transforms encoded audio data received from the network back into a form suitable for reproduction, for example on the terminal's loudspeaker. The output of the audio codec is passed to a delay block **87**. This compensates for the delays introduced by the video coding process and thus ensures synchronisation of audio and video content.

The system control block **84** of the multimedia terminal controls end-to-network signalling using an appropriate control protocol (signalling block **88**) to establish a common mode of operation between a transmitting and a receiving terminal. The signalling block **88** exchanges information about the encoding and decoding capabilities of the transmitting and receiving terminals and can be used to enable the various coding modes of the video encoder. The system control block **84** also controls the use of data encryption. Information regarding the type of encryption to be used in data transmission is passed from encryption block **89** to the multiplexer/demultiplexer (MUX/DMUX unit) **90**.

During data transmission from the multimedia terminal, the MUX/DMUX unit **90** combines encoded and synchronised video and audio streams with data input from the telematic equipment **83** and possible control data, to form a single bit-stream. Information concerning the type of data encryption (if any) to be applied to the bit-stream, provided by encryption block **89**, is used to select an encryption mode. Correspondingly, when a multiplexed and possibly encrypted multimedia bit-stream is being received, MUX/DMUX unit **90** is responsible for decrypting the bit-stream, dividing it into its constituent multimedia components and passing those components to the appropriate codec(s) and/or terminal equipment for decoding and reproduction.

If the multimedia terminal **80** is a mobile terminal, that is, if it is equipped with a radio transceiver **93**, it will be understood by those skilled in the art that it may also comprise additional elements. In one embodiment it comprises a user interface having a display and a keyboard, which enables operation of the multimedia terminal **80** by a user, a central processing unit, such as a microprocessor, which controls the blocks responsible for different functions of the multimedia terminal, a random access memory RAM, a read only memory ROM, and a digital camera. The microprocessor's operating instructions, that is program code corresponding to the basic functions of the multimedia terminal **80**, is stored in the read-only memory ROM and can be executed as required

by the microprocessor, for example under control of the user. In accordance with the program code, the microprocessor uses the radio transceiver **93** to form a connection with a mobile communication network, enabling the multimedia terminal **80** to transmit information to and receive information from the mobile communication network over a radio path.

The microprocessor monitors the state of the user interface and controls the digital camera. In response to a user command, the microprocessor instructs the camera to record digital images into the RAM. Once an image or digital video sequence is captured, or alternatively during the capturing process, the microprocessor segments the image into image segments (for example macroblocks) and uses the encoder to perform motion compensated encoding of the segments in order to generate a compressed image sequence, as explained in the foregoing description. A user may command the multimedia terminal **80** to display the captured images on its display or to send the compressed video sequence using the radio transceiver **93** to another multimedia terminal, a video telephone connected to a fixed line network (PSTN) or some other telecommunications device. In a preferred embodiment, transmission of image data is started as soon as the first segment is encoded so that the recipient can start a corresponding decoding process with a minimum delay.

Although described in the context of particular embodiments, it will be apparent to those skilled in the art that a number of modifications and various changes to these teachings may occur. Thus, while the invention has been particularly shown and described with respect to one or more preferred embodiments thereof, it will be understood by those skilled in the art that certain modifications or changes may be made therein without departing from the scope and spirit of the invention as set forth above.

In particular, according to an alternative embodiment of the invention, surrounding motion analysis block **802** is adapted to classify the motion of a surrounding region into more than two motion classes. For example, one meaningful classification involving three classes of motion would be "continuous motion", "active motion" and "non-active motion". In this way special motion parameters for the typical case of continuous motion can be generated.

In another alternative embodiment of the invention, the surrounding motion analysis block is removed and active motion parameter generation block is activated for all the skip mode macroblocks.

According to a further alternative embodiment Instead of using the surrounding motion analysis block to indicate the classification information, the indication is provided by other means (for example as side information in macroblock, slice, picture or sequence levels).

In yet another alternative embodiment, the surrounding motion analysis block may be temporarily disabled or enabled with such means.

In another alternative implementation, the macroblock mode table is rehashed depending on the output of the surrounding motion analysis block to give higher priority to more likely modes. In a related embodiment, the macroblock mode table is completely regenerated depending on the output of the surrounding motion analysis block, for example, by removing the skip.

It should also be appreciated that active motion parameters generation block **803** can be implemented in various ways. In particular embodiments of the invention it is adapted to generate the motion parameters, for example, based on the continuance, velocity or deviation of the surrounding motion. Additional side information can also be sent to guide the generation of motion parameters. In an alternative embodi-

25

ment the active motion parameter generation block is adapted to output multiple motion vectors to be used in different parts of the macroblock.

The invention can also be used to generate motion compensated prediction for other Macroblock modes in addition to or instead of the SKIP mode. It is also not limited by the macroblock structure but can be used in any segmentation based video coding system.

What is claimed is:

1. A method of encoding a video sequence, the method comprising:

assigning a skip coding mode to a first segment of a first frame of the sequence;

assigning either a zero motion vector or a predicted non-zero motion vector for the skip coding mode for the first segment based at least in part on the motion information of a second segment neighboring the first segment; and forming a prediction for the first segment with respect to a reference frame based at least in part on the assigned motion vector for the skip coding mode, wherein the assigned motion vector is one of the zero motion vector and the predicted non-zero motion vector; and

providing in an encoded bitstream an indication of the skip coding mode, wherein no further motion vector information for the first segment is coded in the encoded bitstream.

2. A method according to claim 1, wherein the second segment is a previously encoded segment neighboring the first segment.

3. A method according to claim 1, wherein if the motion in the second segment has an insignificant level of motion, the zero motion vector is assigned to the skip coding mode for the first segment and the prediction for the first segment is formed with respect to a corresponding segment of the reference frame associated with the zero motion vector.

4. A method according to claim 1, wherein if the motion in the second segment has a motion characteristic of a global or a regional motion,

the method further comprising:

deriving the predicted non-zero motion vector based at least in part on the motion vector of the second segment; and

the predicted non-zero motion vector is assigned to the skip coding mode of the first segment and the prediction for the first segment is formed by a motion compensated prediction with respect to the reference frame based at least in part on the predicted non-zero motion vector.

5. A method according to claim 1, further comprising:

performing an analysis of motion of a region surrounding the first segment;

if it is determined that the region surrounding the first segment has at least one of the following types of motion: continuous motion, motion having a common velocity, and motion having a certain deviation,

the method further comprising deriving the predicted non-zero motion vector based at least in part on the motion vector of the second segment; and

the predicted non-zero motion vector is assigned for the skip coding mode of the first segment, and the prediction for the first segment is formed by a motion compensated prediction with respect to the reference frame based at least in part on the predicted non-zero motion vector; and

otherwise, the zero motion vector is assigned for the skip coding mode of the first segment, and the prediction for

26

the first segment is formed with respect to a corresponding segment of the reference frame associated with the zero motion vector.

6. A method according to claim 1, wherein if the second segment has zero motion vector, the zero motion vector is assigned for skip coding mode of the first segment and the prediction for the first segment is formed with respect to a corresponding segment of the reference frame associated with the zero motion vector.

7. A method of decoding an encoded video sequence, the method comprising:

receiving an indication of a skip coding mode for a first segment;

assigning either a zero motion vector or a predicted non-zero motion vector for the skip coding mode for the first segment based at least in part on the motion information of a second segment neighboring the first segment; and forming a prediction for the first segment with respect to a reference frame based at least in part on the assigned motion vector for the skip coding mode, wherein the assigned motion vector is one of the zero motion vector and the predicted non-zero motion vector.

8. A method according to claim 7, further comprising:

performing an analysis of motion of a region surrounding the first segment;

if it is determined that the region surrounding the first segment has at least one of the following types of motion: continuous motion, motion having a common velocity, and motion having a certain deviation,

the method further comprising deriving the predicted non-zero motion vector based at least in part on the motion vector of the second segment; and

the predicted non-zero motion vector is assigned for the skip coding mode of the first segment; and the prediction for the first segment is formed by a motion compensated prediction with respect to the reference frame based at least in part on the predicted non-zero motion vector; and

otherwise, the zero motion vector is assigned for the skip coding mode of the first segment, and the prediction for the first segment is formed with respect to a corresponding segment of the reference frame associated with the zero motion vector.

9. A method according to claim 7, wherein if a segment in a previously decoded region surrounding the first segment has zero motion vector, the zero-motion vector is assigned to the skip coding mode of the first segment and the prediction for the first segment is formed with respect to a corresponding segment of the reference frame associated with the zero motion vector.

10. A video encoder for encoding a video sequence, the encoder comprises:

a coding controller for assigning a skip coding mode to a first segment;

a motion estimation block for

assigning either a zero motion vector or a predicted non-zero motion vector for the skip coding mode for the first segment based at least in part on the motion information of a second segment neighboring the first segment; and

forming a prediction for the first segment with respect to a reference frame based at least in part on the assigned motion vector for the skip coding mode, wherein the assigned motion vector is one of the zero motion vector and the predicted non-zero motion vector; and

27

a multiplexer for providing in an encoded bitstream an indication of the skip coding mode, wherein no further motion vector information for the first segment is coded in the encoded bitstream.

11. An encoder according to claim 10, wherein the second segment is a previously encoded segment neighboring the first segment.

12. An encoder according to claim 10, wherein if the second segment has an insignificant level of motion, the motion estimation block is arranged to assign the zero motion vector for the skip coding mode of the first segment and the prediction for the first segment is formed with respect to a corresponding segment of the reference frame associated with the zero motion vector.

13. An encoder according to claim 10, wherein if the second segment has a motion characteristic of a global or a regional motion,

the motion estimation block is arranged to derive the predicted non-zero motion vector based at least in part on the motion vector of the second segment; and

the predicted non-zero motion vector is assigned to the skip coding mode of the first segment and the prediction for the first segment is formed by a motion compensated prediction with respect to the reference frame based at least in part on the predicted non-zero motion vector.

14. An encoder according to claim 10, wherein the motion estimation block is arranged to perform an analysis of motion of a region surrounding the first segment;

if it is determined that the region surrounding the first segment has at least one of the following types of motion: continuous motion, motion having a common velocity, and motion having a certain deviation

the motion estimation block is further arranged to derive a predicted non-zero motion vector based at least in part on the motion vector of the second segment; and the predicted non-zero motion vector is assigned for the skip coding mode of the first segment, and the prediction for the first segment is formed by a motion compensated prediction with respect to the reference frame based at least in part on the predicted non-zero motion vector; and

otherwise, the zero motion vector is assigned for the skip coding mode of the first segment, and the prediction for the first segment is formed with respect to a corresponding segment of the reference frame associated with the zero motion vector.

15. An encoder according to claim 10, wherein if a segment in a region surrounding the first segment has zero motion vector, the encoder is arranged to assign the zero motion vector for the skip coding mode of the first segment and the prediction for the first segment is formed with respect to a corresponding segment of the reference frame associated with the zero motion vector.

16. A video decoder for decoding an encoded video sequence, the decoder comprising:

a demultiplexer for receiving an indication of a skip coding mode assigned to a first segment;

a motion compensated prediction block for

assigning either a zero motion vector or a predicted non-zero motion vector for the skip coding mode for the first segment based at least in part on the motion information of a second segment neighboring the first segment; and

forming a prediction for the first segment with respect to a reference frame based at least in part on the assigned motion vector for the skip coding mode, wherein the

28

assigned motion vector is one of the zero motion vector and the predicted non-zero motion vector.

17. A decoder according to claim 16, wherein if the second segment has an insignificant level of motion, the decoder is arranged to assign the zero-motion vector to the skip coding mode for the first segment and the prediction for the first segment is formed with respect to a corresponding segment of the reference frame associated with the zero motion vector.

18. A decoder according to claim 16, wherein if the second segment has a motion characteristic of a global or a regional motion, the decoder is arranged to derive the predicted non-zero motion vector based at least in part on the motion vector of the second segment; and

the predicted non-zero motion vector is assigned to the skip coding mode of the first segment and the prediction for the first segment is formed by a motion compensated prediction with respect to the reference frame based at least in part on the predicted non-zero motion vector.

19. A decoder according to claim 16, wherein the motion compensated prediction block is configured to perform an analysis of motion of a region surrounding the first segment; and

if it is determined that the region surrounding the first segment has at least one of the following types of motion: continuous motion, motion having a common velocity, motion having a certain deviation,

the motion compensated prediction block further configured to derive a predicted non-zero motion vector based at least in part on the motion vector of the second segment; and

the predicted non-zero motion vector is assigned for the skip coding mode of the first segment, and the prediction for the first segment is formed by a motion compensated prediction with respect to the reference frame based at least in part on the predicted non-zero motion vector; and

otherwise, the zero motion vector is assigned for the skip coding mode of the first segment, and the prediction for the first segment is formed with respect to a corresponding segment of the reference frame associated with the zero motion vector.

20. A decoder according to claim 16, wherein if a segment in a region surrounding the first segment has zero motion vector, the zero motion vector is assigned for the skip coding mode of the first segment, and the prediction for the first segment is formed with respect to a corresponding segment of the reference frame associated with the zero motion vector.

21. A multimedia terminal, comprising an encoder according to claim 10.

22. A multimedia terminal comprising a decoder according to claim 16.

23. A method according to claim 1, wherein if the second segment has a non-zero motion vector,

the method further comprising deriving the predicted non-zero motion vector based at least in part on the motion vector of the second segment; and

the predicted non-zero motion vector is assigned for the skip coding mode of the first segment; and the prediction for the first segment is formed by a motion compensated prediction with respect to the reference frame based at least in part on the predicted non-zero motion vector.

24. A method according to claim 1, wherein if the second segment has a zero motion vector and the second segment is predicted using motion-compensated prediction from the reference picture, the zero motion vector is assigned to the skip coding mode of the first segment and the prediction for the

first segment is formed with respect to a corresponding segment of the reference frame associated with the zero motion vector.

25. A method according to claim 1, wherein if the second segment has a zero motion vector and the second segment is predicted using motion-compensated prediction from a second reference picture immediately preceding the picture second segment belongs to, the zero motion vector is assigned to the skip coding mode of the first segment and the prediction for the first segment is formed with respect to a corresponding segment of the reference frame associated with the zero motion vector.

26. A method according to claim 1, further comprising: performing an analysis of motion information of the second segment and motion information of a third segment neighboring the first segment; and

determining whether a region surrounding the first segment has a global or a regional motion in a video sequence based at least in part on a characteristic of the motion vector of the second segment and the motion vector of the third segment.

27. A method according to claim 26, wherein if the region surrounding the first segment has a global or a regional motion in a video sequence,

the method further comprising deriving the predicted non-zero motion vector based at least in part on the motion vector of the second segment and the motion vector of the third segment; and

the predicted non-zero motion vector is assigned for the skip coding mode of the first segment; and the prediction for the first segment is formed by a motion compensated prediction with respect to the reference frame based at least in part on the predicted non-zero motion vector.

28. A method according to claim 1, further comprising: deriving the predicted non-zero motion vector based at least in part on the motion vector of the second segment.

29. A method according to claim 1, further comprising: deriving the predicted non-zero motion vector based at least in part on the motion vector of the second segment and motion vector of a third segment neighboring the first segment.

30. A method according to claim 1, wherein no residual information is provided for the first segment in the encoded bitstream.

31. A method according to claim 1, further comprising: deriving a predicted motion vector based at least in part on the motion vector of the second segment and motion vector of a third segment neighboring the first segment; and

if any component of the predicted motion vector has an absolute value larger than a certain threshold value, the predicted motion vector is assigned for the skip coding mode of the first segment, and the prediction for the first segment is formed by a motion compensated prediction with respect to the reference frame based at least in part on the predicted motion vector; and

if none of components of the predicted motion vector has an absolute value larger than the certain threshold value, the zero motion vector is assigned for the skip coding mode of the first segment, and the prediction for the first segment is formed with respect to a corresponding segment of the reference frame associated with the zero motion vector.

32. A method according to claim 7, wherein if the second segment has a zero motion vector, the zero motion vector is assigned to the skip coding mode of the first segment and the

prediction for the first segment is formed with respect to a corresponding segment of the reference frame associated with the zero motion vector.

33. A method according to claim 7, wherein if the second segment has an insignificant level of motion, the zero motion vector is assigned to the skip coding mode of the first segment and the prediction for the first segment is formed with respect to a corresponding segment of the reference frame associated with the zero motion vector.

34. A method according to claim 7, wherein if the second segment has a non-zero motion vector,

the method further comprising deriving the predicted non-zero motion vector based at least in part on the motion vector of the second segment; and

the predicted non-zero motion vector is assigned for the skip coding mode of the first segment; and the prediction for the first segment is formed by a motion compensated prediction with respect to the reference frame based at least in part on the predicted non-zero motion vector.

35. A method according to claim 7, characterized in that if the second segment has a motion characteristic of a global or a regional motion,

the method further comprising deriving the predicted non-zero motion vector based at least in part on the motion vector of the second segment; and

the predicted non-zero motion vector is assigned to the skip coding mode of the first segment and the prediction for the first segment is formed by a motion compensated prediction with respect to the reference frame based at least in part on the predicted non-zero motion vector.

36. A method according to claim 7, wherein if the second segment has a zero motion vector and the second segment is predicted using motion-compensated prediction from the reference picture, the zero motion vector is assigned to the skip coding mode of the first segment and the prediction for the first segment is formed with respect to a corresponding segment of the reference frame associated with the zero motion vector.

37. A method according to claim 7, further comprising: performing an analysis of motion information of the second segment and motion information of a third segment neighboring the first segment;

determining whether a region surrounding the first segment has a global or a regional motion in a video sequence based at least in part on a characteristic of the motion vector of the second segment and the motion vector of the third segment.

38. A method according to claim 37, wherein if the region surrounding the first segment has a global or a regional motion in a video sequence,

the method further comprising deriving the predicted non-zero motion vector based at least in part on the motion vector of the second segment and the motion vector of the third segment; and

the predicted non-zero motion vector is assigned for the skip coding mode of the first segment; and the prediction for the first segment is formed by a motion compensated prediction with respect to the reference frame based at least in part on the predicted non-zero motion vector.

39. A method according to claim 7, further comprising: deriving the predicted non-zero motion vector based at least in part on the motion vector of the second segment.

40. A method according to claim 7, further comprising: deriving the predicted non-zero motion vector based at least in part on the motion vector of the second segment and motion vector of a third segment neighboring the first segment.

31

41. A method according to claim 7, wherein no residual information is provided for the first segment in the encoded bitstream.

42. A method according to claim 7, further comprising:
 deriving a predicted motion vector based at least in part on
 the motion vector of the second segment and motion
 vector of a third segment neighboring the first segment;
 and

if any component of the predicted motion vector has an
 absolute value larger than a certain threshold value, the
 predicted motion vector is assigned for the skip coding
 mode of the first segment, and the prediction for the first
 segment is formed by a motion compensated prediction
 with respect to the reference frame based at least in part
 on the predicted motion vector; and

if none of components of the predicted motion vector has
 an absolute value larger than the certain threshold value,
 the zero motion vector is assigned for the skip coding
 mode of the first segment, and the prediction for the first
 segment is formed with respect to a corresponding seg-
 ment of the reference frame associated with the zero
 motion vector.

43. A method according to claim 1, wherein if the second
 segment has a zero motion vector and the second segment is
 predicted using motion-compensated prediction from a sec-
 ond reference picture immediately preceding the picture sec-
 ond segment belongs to, the zero motion vector is assigned to
 the skip coding mode of the first segment and the prediction
 for the first segment is formed with respect to a corresponding
 segment of the reference frame associated with the zero
 motion vector.

44. An encoder according to claim 10, wherein if the sec-
 ond segment has a non-zero motion vector,

the motion estimation block is further arranged to derive
 the predicted non-zero motion vector based at least in
 part on the motion vector of the second segment; and

the predicted non-zero motion vector is assigned for the
 skip coding mode for the first segment; and the predic-
 tion for the first segment is formed by a motion compen-
 sated prediction with respect to the reference frame
 based at least in part on the predicted non-zero motion
 vector.

45. An encoder according to claim 10, further wherein the
 motion estimation block is further arranged to:

perform an analysis of motion information of the second
 segment and motion information of a third segment
 neighboring the first segment;

determine whether a region surrounding the first segment
 has a global or a regional motion in a video sequence
 based at least in part on a characteristic of the motion
 vector of the second segment and the motion vector of
 the third segment.

46. An encoder according to claim 45, wherein if the region
 surrounding the first segment has a global or a regional
 motion in a video sequence,

the motion estimation block further arranged to derive the
 predicted non-zero motion vector based at least in part
 on the motion vector of the second segment and the
 motion vector of the third segment; and

the predicted non-zero motion vector is assigned for the
 skip coding mode for the first segment; and the predic-
 tion for the first segment is formed by a motion compen-
 sated prediction with respect to the reference frame
 based at least in part on the predicted non-zero motion
 vector.

32

47. An encoder according to claim 10, wherein the motion
 estimation block further arranged to derive the predicted non-
 zero motion vector based at least in part on the motion vector
 of the second segment.

48. An encoder according to claim 10, wherein the motion
 estimation block further arranged to derive the predicted non-
 zero motion vector based at least in part on the motion vector
 of the second segment and motion vector of a third segment
 neighboring the first segment.

49. An encoder according to claim 10, wherein no residual
 information is provided for the first segment in the encoded
 bitstream.

50. An encoder according to claim 10, wherein the motion
 estimation block further arranged to derive a predicted
 motion vector based at least in part on the motion vector of the
 second segment and motion vector of a third segment neigh-
 boring the first segment; and

if any component of the predicted motion vector has an
 absolute value larger than a certain threshold value, the
 predicted motion vector is assigned for the skip coding
 mode of the first segment, and the prediction for the first
 segment is formed by a motion compensated prediction
 with respect to the reference frame based at least in part
 on the predicted motion vector; and

if none of components of the predicted motion vector has
 an absolute value larger than the certain threshold value,
 the zero motion vector is assigned for the skip coding
 mode of the first segment, and the prediction for the first
 segment is formed with respect to a corresponding seg-
 ment of the reference frame associated with the zero
 motion vector.

51. An encoder according to claim 10, wherein if the sec-
 ond segment has a zero motion vector, the zero motion vector
 is assigned to the skip coding mode of the first segment and
 the prediction for the first segment is formed with respect to a
 corresponding segment of the reference frame associated
 with the zero motion vector.

52. An encoder according to claim 10, wherein if the sec-
 ond segment has a zero motion vector and the second segment
 is predicted using motion-compensated prediction from the
 reference picture, the zero motion vector is assigned to the
 skip coding mode of the first segment and the prediction for
 the first segment is formed with respect to a corresponding
 segment of the reference frame associated with the zero
 motion vector.

53. An encoder according to claim 10, wherein if the sec-
 ond segment has a zero motion vector and the second segment
 is predicted using motion-compensated prediction from a
 second reference picture immediately preceding the picture
 second segment belongs to, the zero motion vector is assigned
 to the skip coding mode of the first segment and the prediction
 for the first segment is formed with respect to a corresponding
 segment of the reference frame associated with the zero
 motion vector.

54. A decoder according to claim 16, wherein if the second
 segment has a zero motion vector, the zero-motion vector is
 assigned to the skip coding mode for the first segment and the
 prediction for the first segment is formed with respect to a
 corresponding segment of the reference frame associated
 with the zero motion vector.

55. A decoder according to claim 16, wherein if the second
 segment has a non-zero motion vector,

the motion compensated prediction block further arranged
 to derive the predicted non-zero motion vector based at
 least in part on the motion vector of the second segment;
 and

33

the predicted non-zero motion vector is assigned for the skip coding mode for the first segment; and the prediction for the first segment is formed by a motion compensated prediction with respect to the reference frame based at least in part on the predicted non-zero motion vector. 5

56. A decoder according to claim 16, wherein the motion estimation block further arranged to:

perform an analysis of motion information of the second segment and motion information of a third segment neighboring the first segment; 10

determine whether a region surrounding the first segment has a global or a regional motion in a video sequence based at least in part on a characteristic of the motion vector of the second segment and the motion vector of the third segment. 15

57. A decoder according to claim 56, wherein if the region surrounding the first segment has a global or a regional motion in a video sequence,

the motion estimation block further arranged to derive the predicted non-zero motion vector based at least in part on the motion vector of the second segment and the motion vector of the third segment; and 20

the predicted non-zero motion vector is assigned for the skip coding mode for the first segment; and the prediction for the first segment is formed by a motion compensated prediction with respect to the reference frame based at least in part on the predicted non-zero motion vector. 25

58. A decoder according to claim 16, wherein the motion estimation block further arranged to derive the predicted non-zero motion vector based at least in part on the motion vector of the second segment. 30

59. A decoder according to claim 16, wherein the motion estimation block further arranged to derive the predicted non-zero motion vector based at least in part on the motion vector of the second segment and motion vector of a third segment neighboring the first segment. 35

60. A decoder according to claim 16, wherein no residual information is provided for the first segment in an encoded bitstream. 40

61. A decoder according to claim 16, wherein the motion estimation block further arranged to derive a predicted

34

motion vector based at least in part on the motion vector of the second segment and motion vector of a third segment neighboring the first segment; and

if any component of the predicted motion vector has an absolute value larger than a certain threshold value, the predicted motion vector is assigned for the skip coding mode of the first segment, and the prediction for the first segment is formed by a motion compensated prediction with respect to the reference frame based at least in part on the predicted motion vector; and

if none of components of the predicted motion vector has an absolute value larger than the certain threshold value, the zero motion vector is assigned for the skip coding mode of the first segment, and the prediction for the first segment is formed with respect to a corresponding segment of the reference frame associated with the zero motion vector.

62. A decoder according to claim 16, wherein if the second segment has a zero motion vector and the second segment is predicted using motion-compensated prediction from the reference picture, the zero motion vector is assigned to the skip coding mode of the first segment and the prediction for the first segment is formed with respect to a corresponding segment of the reference frame associated with the zero motion vector.

63. A decoder according to claim 16, wherein if the second segment has a zero motion vector and the second segment is predicted using motion-compensated prediction from a second reference picture immediately preceding the picture second segment belongs to, the zero motion vector is assigned to the skip coding mode of the first segment and the prediction for the first segment is formed with respect to a corresponding segment of the reference frame associated with the zero motion vector.

64. A decoder according to claim 16, wherein no further motion vector information for the first segment is retrieved from an encoded bitstream.

65. A method according to claim 7, wherein no further motion vector information for the first segment is retrieved from the encoded bitstream.

* * * * *

EXHIBIT 5



US008050321B2

(12) **United States Patent**
Hannuksela

(10) **Patent No.:** **US 8,050,321 B2**
(45) **Date of Patent:** **Nov. 1, 2011**

(54) **GROUPING OF IMAGE FRAMES IN VIDEO CODING**

(75) Inventor: **Miska Hannuksela**, Tampere (FI)

(73) Assignee: **Nokia Corporation**, Espoo (FI)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1632 days.

(21) Appl. No.: **11/338,934**

(22) Filed: **Jan. 25, 2006**

(65) **Prior Publication Data**

US 2006/0120451 A1 Jun. 8, 2006

Related U.S. Application Data

(63) Continuation of application No. 10/306,942, filed on Nov. 29, 2002, now Pat. No. 7,894,521.

(30) **Foreign Application Priority Data**

Jan. 23, 2002 (FI) 20020127

(51) **Int. Cl.**
H04N 7/12 (2006.01)

(52) **U.S. Cl.** **375/240.12; 375/240.25**

(58) **Field of Classification Search** **375/240.12, 375/240.25**

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,390,966 A * 6/1983 Kawashima et al. 712/229
5,122,875 A 6/1992 Raychaudhuri et al.
5,144,426 A 9/1992 Tanaka et al.
5,680,322 A * 10/1997 Shinoda 714/18
5,699,476 A 12/1997 Van Der Meer
5,774,593 A 6/1998 Zick et al.
5,786,858 A * 7/1998 Yagasaki et al. 375/240.15

5,838,265 A 11/1998 Adolph
5,852,630 A * 12/1998 Langberg et al. 375/219
5,877,812 A 3/1999 Krause et al.
6,072,831 A 6/2000 Chen
6,094,456 A * 7/2000 Ueda 375/240.12
6,108,382 A 8/2000 Gringeri et al.
6,266,158 B1 7/2001 Hata et al.

(Continued)

FOREIGN PATENT DOCUMENTS

CN 1173783 A 2/1998

(Continued)

OTHER PUBLICATIONS

2010, English Translation of the Office Action of parallel Japanese application No. 2006-120296 dated Jan. 6, 2010.

(Continued)

Primary Examiner — Shuwang Liu

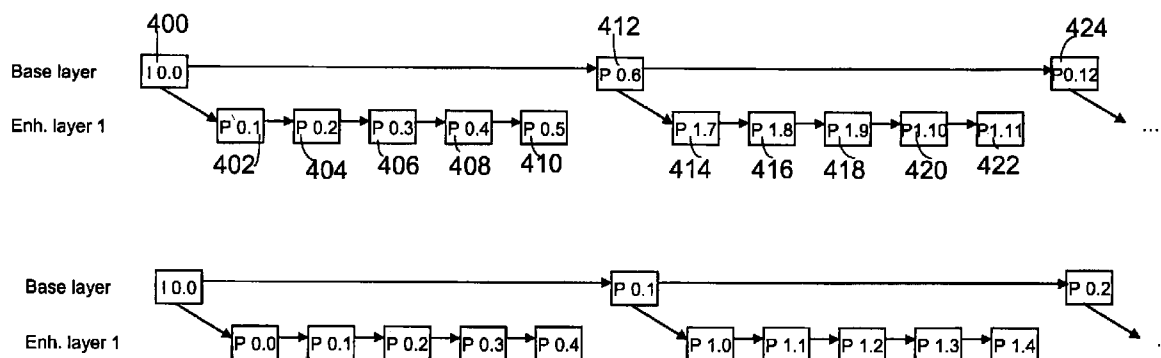
Assistant Examiner — Siu Lee

(74) *Attorney, Agent, or Firm* — Hollingsworth & Funk, LLC

(57) **ABSTRACT**

A method for encoding a video sequence comprising an independent sequence of image frames, wherein at least one reference image frame is predictable from at least one previous image frame that is earlier than the previous reference image frame in decoding order. An indication of at least one image frame is encoded into the video sequence, which indicated image frame is the first image frame, in decoding order, of the independent sequence, said at least one reference image frame being included in the sequence. In the decoding phase, the indication of at least one image frame is decoded from the video sequence, and the decoding of the video sequence is started from said first image frame of the independent sequence, whereby the video sequence is decoded without prediction from any image frame decoded prior to said first image frame.

11 Claims, 4 Drawing Sheets



U.S. PATENT DOCUMENTS

6,307,886	B1	10/2001	Westermann	
6,314,139	B1	11/2001	Koto et al.	
6,337,881	B1	1/2002	Chaddha	
6,363,208	B2	3/2002	Nitta et al.	
6,483,875	B1 *	11/2002	Hasebe et al.	375/240.15
6,496,980	B1	12/2002	Tillman et al.	
6,510,553	B1	1/2003	Hazra	
6,614,936	B1	9/2003	Wu et al.	
6,639,943	B1	10/2003	Radha et al.	
7,103,669	B2 *	9/2006	Apostolopoulos	709/231
2001/0024472	A1	9/2001	Sporer et al.	
2001/0040700	A1 *	11/2001	Hannuksela et al.	358/261.2
2002/0051621	A1 *	5/2002	Cuccia	386/68
2003/0063806	A1 *	4/2003	Kim et al.	382/236

FOREIGN PATENT DOCUMENTS

EP	0798932	10/1997
EP	0898427	2/1999
EP	1 005 232	5/2000
JP	4348690	12/1992
JP	8265694	10/1996
JP	10-271511	10/1998
JP	11-177987	2/1999
JP	11275576	10/1999
RU	2 123 769	12/1998
RU	2137197 C1	9/1999
WO	WO 00/67469	11/2000
WO	WO 01/47283	6/2001
WO	01/84850	11/2001
WO	WO 02/054776	7/2002

OTHER PUBLICATIONS

Elloumi et al., "Issues in Multimedia Scaling: the MPEG Video Streams Case", Emerging Technologies and Applications in Communications, IEEE Computer Society, May 7, 1996.

Hannuksela, "Enhanced Concept of GOP", Joint Video Team of ISO-IES MPEG 7 itu-T VCEG, Feb. 1, 2002.

Soderquist et al., "Memory Traffic and Data Cache Behavior of an MPEG-2 Software Decoder", IEEE Computer Society, Oct. 12, 1997.

Office Action of related European application No. 06101013.8 dated Feb. 2, 2010.

Office Action of related European application No. 06101014.6 dated Feb. 2, 2010.

Office Action of parallel U.S. Appl. No. 10/306,942 dated Mar. 16, 2006, 4 pages.

Office Action of parallel U.S. Appl. No. 10/306,942 dated Aug. 10, 2006, 16 pages.

Office Action of parallel U.S. Appl. No. 10/306,942 dated Sep. 11, 2007, 4 pages.

Office Action of parallel U.S. Appl. No. 10/306,942 dated Mar. 13, 2007, 7 pages.

Notice of Allowance of parallel U.S. Appl. No. 10/306,942 dated Dec. 7, 2009, 8 pages.

May 6, 2006, Office Action Response of parallel U.S. Appl. No. 10/306,942 dated Aug. 6, 2006, 13 pages.

Office Action Response of parallel U.S. Appl. No. 10/306,942 dated Nov. 9, 2006, 16 pages.

Office Action Response of parallel U.S. Appl. No. 10/306,942 dated May 31, 2007, 12 pages.

Office Action Response of parallel U.S. Appl. No. 10/306,942 dated Jul. 9, 2009, 12 pages.

Allowance Notification with translation dated Feb. 4, 2010 from parallel Russian Application No. 2006110322, 17 pages.

Translated Office Action dated Feb. 25, 2010 from parallel Japanese Application No. 2006-120297, 5 pages.

Oct. 9-17, 2002, Joint Video Team of ISO/IEC MPEG and ITU-T VCEG, *Editor's Proposed Draft Text Modification for Joint Video Specification (ITU-T Rec. 264 ISO/IEC 14496-10 AVC)*, Geneva modifications draft 37.

Jan. 15, 1998, ITU-Telecommunications Standardization Union, *Draft Text of Recommendation H.263 Version 2 ("H.263+") for Decision*.

Jul. 10, 2002, ITU-Telecommunications Standardization Union, *H.26L Test Model Long Term No. 8 (TML-8) Draft0*, Document VCEG-N10, pp. 1-46.

Nov. 2000, ITU-Telecommunications Standardization Union, *Draft for "H.263++" Annexes U,V, and W to Recommendation H.263*.

Nov. 12, 2002, David Singer and Toby Walker, *Study Text of ISO/IEC 14496-15/FCD*, pp. 1-25.

Nov. 12, 2002, David Singer and Toby Walker, *Study Text of ISO/IEC 14496-15/FCD (Revised)*, 26 pages.

Sep. 2002, Hannuksela, M et al.; Sub-Picture: ROI Coding and Unequal Error Protection, IEEE 2002 Conference, Sep. 2002, USA (internet publication).

Sep. 2002, Hannuksela, M et al.; Sub-Picture: Video-Coding for Unequal Error Protection, XI European Signal Processing Conference, Sep. 2002, Toulouse, France (internet publication).

Jul. 12, 2001, Wang, Y.K. et al.; Core Experiment Description of Sub-Picture Coding, ITU 15. Meeting Dec. 3-7, 2001, Pattaya, Thailand (internet publication).

2001, B.A. Lokshin, "Digital Broadcasting: from Studio to Audience" Cyrus Systems, Moscow, 2001. (translation included).

Wenger, "Temporal Scalability Using P-pictures for Low-latency Applications", IEEE Workshop, Dec. 1998, pp. 559-564.

1997, Illgner et al., "Spatially Scalable Video Compression Employing Resolution Pyramids", IEEE Journal on Selected Areas in Communications, vol. 15, No. 9, Dec. 1, 2007, pp. 1688-1703.

Information Technology—Generic Coding of Moving Pictures and Associated Audio Information: Systems, ISO/IEC 13818-1, Second edition, Dec. 1, 2000, 174 pages.

Notice of Allowance dated May 5, 2010 from Russian Application No. 2006110321/09, 11 pages.

Office Action dated Jun. 17, 2010 from U.S. Appl. No. 11/338,996, 30 pages.

Office Action dated Jun. 23, 2010 from U.S. Appl. No. 10/306,942, 6 pages.

Office Action dated May 25, 2010 from U.S. Appl. No. 10/306,942, 6 pages.

Office Action dated Apr. 9, 2009 from U.S. Appl. No. 10/306,942, 4 pages.

Office Action Response dated May 8, 2006 from U.S. Appl. No. 10/306,942, 11 pages.

Translated Office Action dated Aug. 17, 2010 from parallel Japanese Application No. 2006-120297, 4 pages.

Translated Office Action dated Aug. 24, 2010 from parallel Japanese Application No. 2006-120296, 5 pages.

Office Action Response dated Sep. 15, 2010 from U.S. Appl. No. 10/306,942, 12 pages.

Office Action Response dated Sep. 10, 2010 from U.S. Appl. No. 11/338,996, 10 pages.

Hannuksela, "Signaling of Enhanced GOPs", Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, 4th Meeting, Austria, Jul. 22-26, 2002.

File History for EP Application No. 03700321.7 as retrieved from the European Patent Office Electronic File System on Apr. 5, 2011, 222 pages.

Office Action dated Feb. 18, 2011 from U.S. Appl. No. 11/338,996, 18 pages.

Office Action Response dated May 17, 2011 from U.S. Appl. No. 11/338,996, 9 pages.

Notice of Allowance dated Aug. 2, 2011 from Japanese Application No. 2006-120296, 3 pages.

* cited by examiner

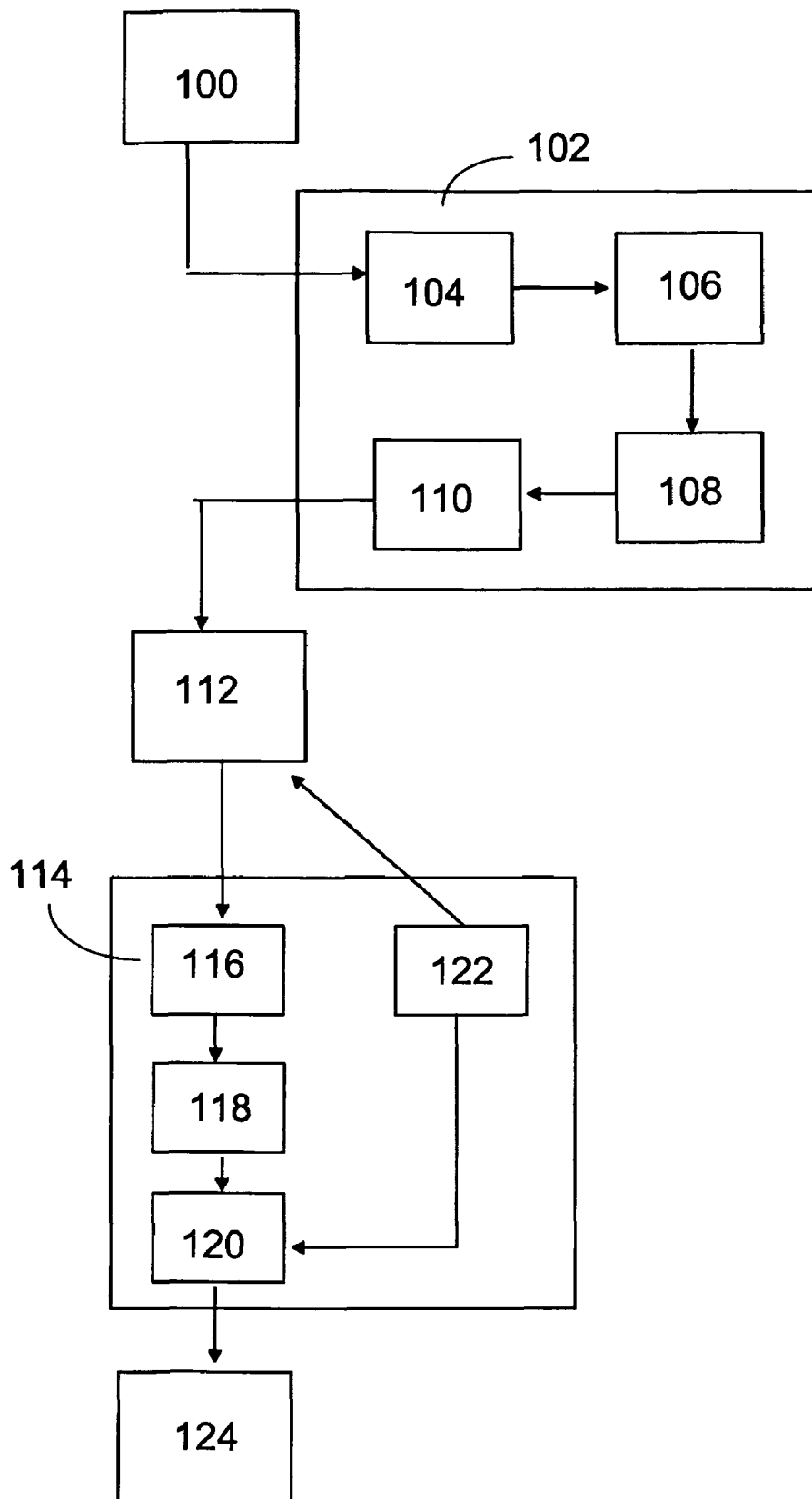


FIG. 1

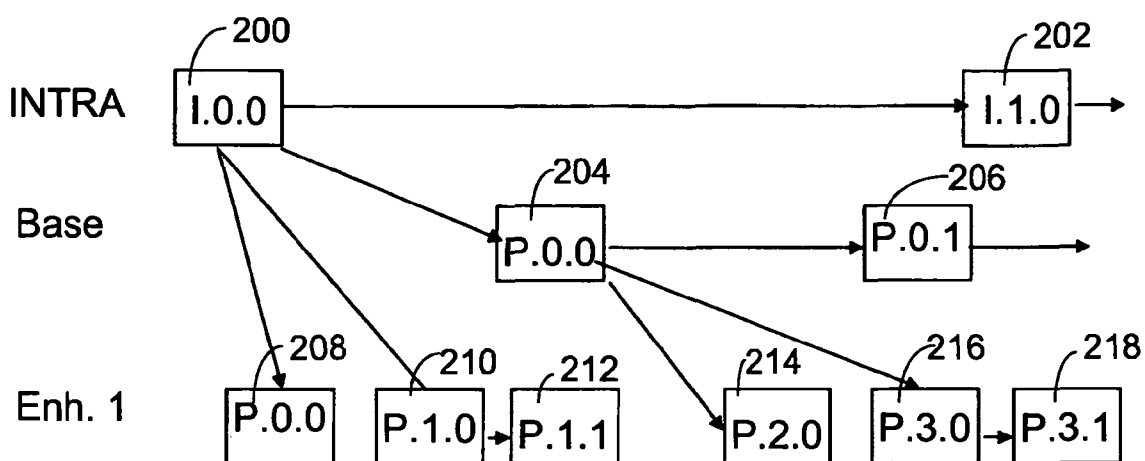


FIG. 2

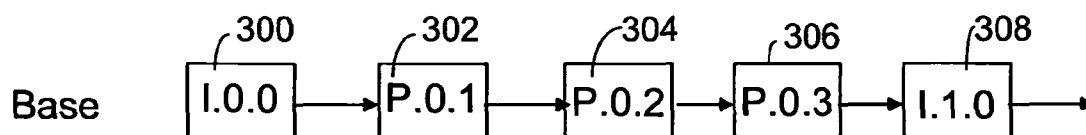


FIG. 3a

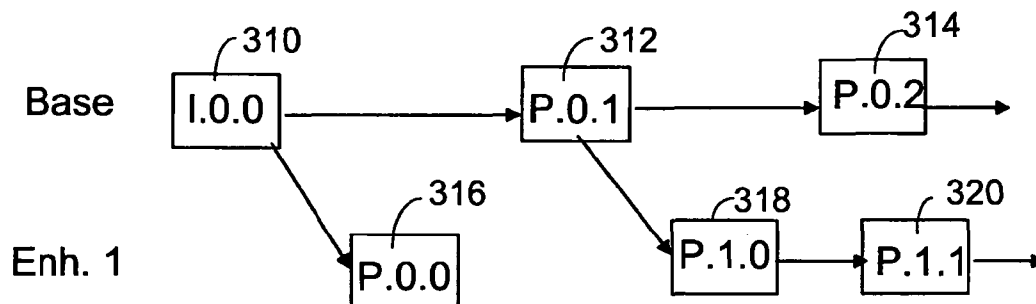


FIG. 3b

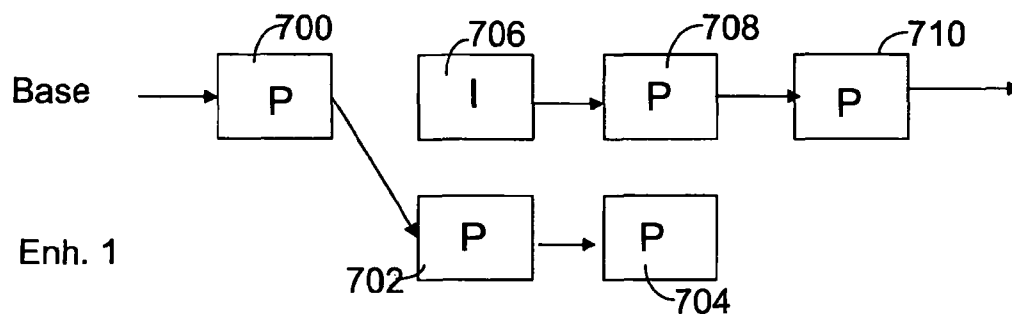


FIG. 7

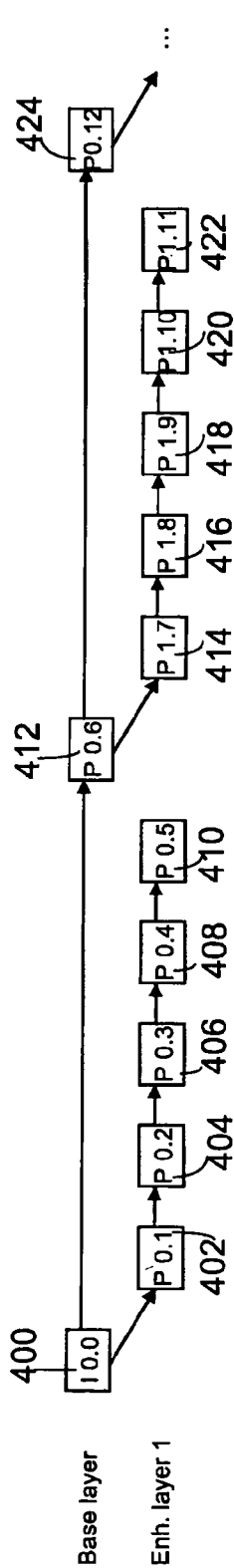


FIG. 4a

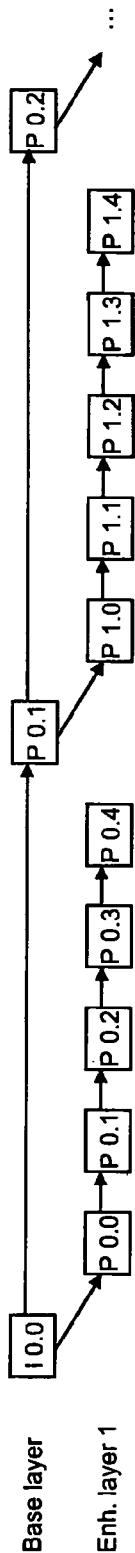


FIG. 4b

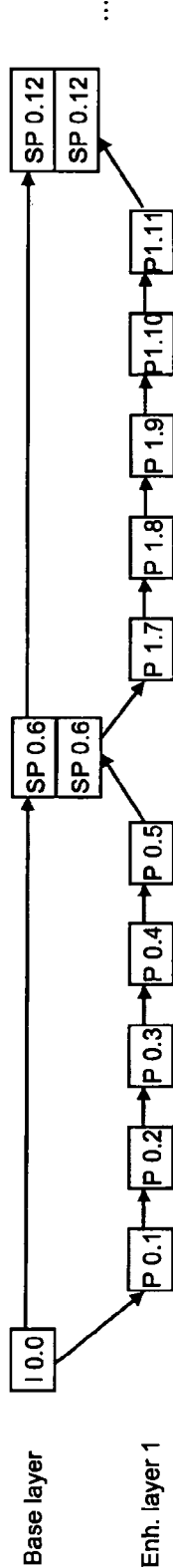


FIG. 4c

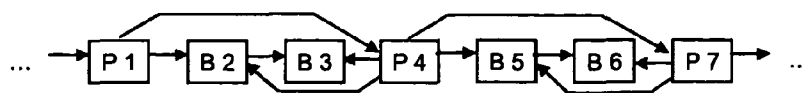


FIG. 5a

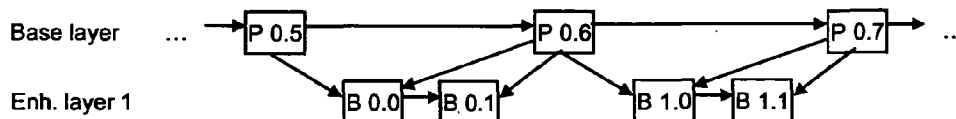


FIG. 5b

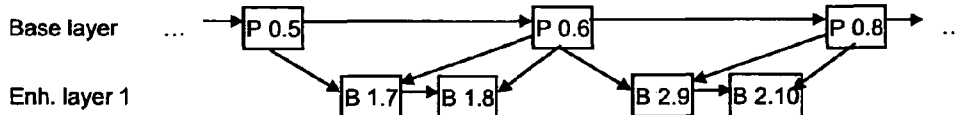


FIG. 5c

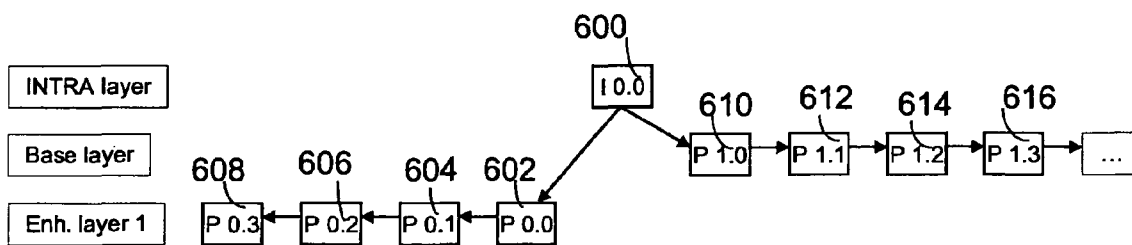


FIG. 6a

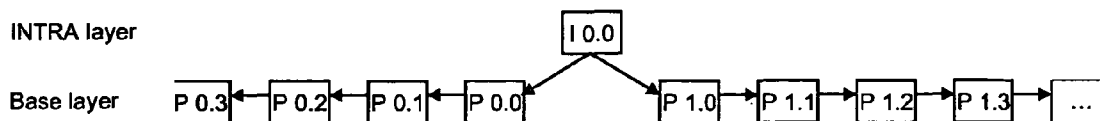


FIG. 6b

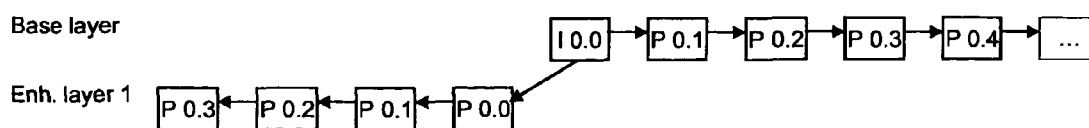


FIG. 6c

1

GROUPING OF IMAGE FRAMES IN VIDEO CODING

RELATED PATENT DOCUMENTS

This application is a continuation of U.S. patent application Ser. No. 10/306,942 filed on Nov. 29, 2002 now U.S. Pat. No. 7,894,521 which is incorporated herein by reference in its entirety.

FIELD OF THE INVENTION

The invention relates to the grouping of multimedia files, particularly video files and particularly in connection with streaming.

BACKGROUND OF THE INVENTION

The term 'streaming' refers to simultaneous sending and playback of data, typically multimedia data, such as audio and video files, in which the recipient may begin data playback already before all the data to be transmitted has been received. Multimedia data streaming systems comprise a streaming server and terminal devices that the recipients use for setting up a data connection, typically via a telecommunications network, to the streaming server. From the streaming server the recipients retrieve either stored or real-time multimedia data, and the playback of the multimedia data can then begin, most advantageously almost in real-time with the transmission of the data, by means of a streaming application included in the terminal.

From the point of view of the streaming server, the streaming may be carried out either as normal streaming or as progressive downloading to the terminal. In normal streaming the transmission of the multimedia data and/or the data contents are controlled either by making sure that the bit rate of the transmission substantially corresponds to the playback rate of the terminal device, or, if the telecommunications network used in the transmission causes a bottleneck in data transfer, by making sure that the bit rate of the transmission substantially corresponds to the bandwidth available in the telecommunications network. In progressive downloading the transmission of the multimedia data and/or the data contents do not necessarily have to be interfered with at all, but the multimedia files are transmitted as such to the recipient, typically by using transfer protocol flow control. The terminals then receive, store and reproduce an exact copy of the data transmitted from the server, which copy can then be later reproduced again on the terminal without needing to start a streaming again via the telecommunications network. The multimedia files stored in the terminal are, however, typically very large and their transfer to the terminal is time-consuming, and they require a significant amount of storage memory capacity, which is why a normal streaming is often preferred.

The video files in multimedia files comprise a great number of still image frames, which are displayed rapidly in succession (of typically 15 to 30 frames per s) to create an impression of a moving image. The image frames typically comprise a number of stationary background objects, determined by image information which remains substantially unchanged, and few moving objects, determined by image information that changes to some extent. The information comprised by consecutively displayed image frames is typically largely similar, i.e. successive image frames comprise a considerable amount of redundancy. The redundancy appearing in video files can be divided into spatial, temporal and spectral redundancy. Spatial redundancy refers to the mutual correlation of

2

adjacent image pixels, temporal redundancy refers to the changes taking place in specific image objects in subsequent frames, and spectral redundancy to the correlation of different colour components within an image frame.

To reduce the amount of data in video files, the image data can be compressed into a smaller form by reducing the amount of redundant information in the image frames. In addition, while encoding, most of the currently used video encoders downgrade image quality in image frame sections that are less important in the video information. Further, many video coding methods allow redundancy in a bit stream coded from image data to be reduced by efficient, lossless coding of compression parameters known as VLC (Variable Length Coding).

In addition, many video coding methods make use of the above-described temporal redundancy of successive image frames. In that case a method known as motion-compensated temporal prediction is used, i.e. the contents of some (typically most) of the image frames in a video sequence are predicted from other frames in the sequence by tracking changes in specific objects or areas in successive image frames. A video sequence always comprises some compressed image frames the image information of which has not been determined using motion-compensated temporal prediction. Such frames are called INTRA-frames, or I-frames. Correspondingly, motion-compensated video sequence image frames predicted from previous image frames, are called INTER-frames, or P-frames (Predicted). The image information of P-frames is determined using one I-frame and possibly one or more previously coded P-frames. If a frame is lost, frames dependent on it can no longer be correctly decoded.

An I-frame typically initiates a video sequence defined as a Group of Pictures (GOP), the P-frames of which can only be determined on the basis of the I-frame and the previous P-frames of the GOP in question. The next I-frame begins a new group of pictures GOP, the image information comprised by which cannot thus be determined on the basis of the frames of the previous GOP. In other words, groups of pictures are not temporally overlapping, and each group of picture can be decoded separately. In addition, many video compression methods employ bi-directionally predicted B-frames (Bi-directional), which are set between two anchor frames (I- and P-frames, or two P-frames) within a group of pictures GOP, the image information of a B-frame being predicted from both the previous anchor frame and the one succeeding the B-frame. B-frames therefore provide image information of higher quality than P-frames, but typically they are not used as anchor frames, and therefore their removal from the video sequence does not degrade the quality of subsequent images. However, nothing prevents B-frames from being used as anchor frames as well, only in that case they cannot be removed from the video sequence without deteriorating the quality of the frames dependent on them.

Each video frame may be divided into what are known as macroblocks that comprise the colour components (such as Y, U, V) of all pixels of a rectangular image area. More specifically, a macroblock consists of at least one block per colour component, the blocks each comprising colour values (such as Y, U or V) of one colour level in the image area concerned. The spatial resolution of the blocks may differ from that of the macroblocks, for example U- and V-components may be displayed using only half of the resolution of Y-component. Macroblocks can be further grouped into slices, for example, which are groups of macroblocks that are typically selected in the scanning order of the image. Temporal prediction is typi-

3

cally carried out in video coding methods block- or macroblock-specifically, instead of image-frame-specifically.

To allow for flexible streaming of video files, many video coding systems employ scalable coding in which some elements or element groups of a video sequence can be removed without affecting the reconstruction of other parts of the video sequence. Scalability is typically implemented by grouping the image frames into a number of hierarchical layers. The image frames coded into the image frames of the base layer substantially comprise only the ones that are compulsory for the decoding of the video information at the receiving end. The base layer of each group of pictures GOP thus comprises one I-frame and a necessary number of P-frames. One or more enhancement layers can be determined below the base layer, each one of the layers improving the quality of the video coding in comparison with an upper layer. The enhancement layers thus comprise P- or B-frames predicted on the basis of motion-compensation from one or more upper layer images. The frames are typically numbered according to an arithmetical series.

In streaming, transmission bit rate must be controllable either on the basis of the bandwidth to be used or the maximum decoding or bit rate value of the recipient. Bit rate can be controlled either at the streaming server or in some element of the telecommunications network, such as an Internet router or a base station of a mobile communications network. The simplest means for the streaming server to control the bit rate is to leave out B-frames having a high information content from the transmission. Further, the streaming server may determine the number of scalability layers to be transmitted in a video stream, and thus the number of the scalability layers can be changed always when a new group of pictures GOP begins. It is also possible to use different video sequence coding methods. Correspondingly, B-frames, as well as other P-frames of the enhancement layers, can be removed from the bit stream in a telecommunications network element.

The above arrangement involves a number of drawbacks. Many coding methods, such as the coding according to the ITU-T (International Telecommunications Union, Telecommunications Standardization Sector) standard H.263, are familiar with a procedure called reference picture selection. In reference picture selection at least a part of a P-image has been predicted from at least one other image than the one immediately preceding the P-image in the time domain. The selected reference image is signalled in a coded bit stream or in bit stream header fields image-, image-segment- (such as a slice or a group of macroblocks), macroblock-, or block-specifically. The reference picture selection can be generalized such that the prediction can also be made from images temporally succeeding the image to be coded. Further, the reference picture selection can be generalized to cover all temporally predicted frame types, including B-frames. Since it is possible to also select at least one image preceding an I-image that begins a group of pictures GOP as the reference image, a group of pictures employing reference picture selection cannot necessarily be decoded independently. In addition, the adjusting of scalability or coding method in the streaming server or a network element becomes difficult, because the video sequence must be decoded, parsed and buffered for a long period of time to allow any dependencies between different image groups to be detected.

A further problem relates to detection of image frames from which a decoder can start the decoding process. The detection is useful for multiple purposes. For example, an end-user may wish to start browsing a video file from the middle of a video sequence. Another example relates to starting the reception of a broadcast or multicast video transmis-

4

sion from the middle of the video transmission. A third example relates to on-demand streaming from a server and occurs when an end-user wishes to start playback from a certain position of a stream.

BRIEF DESCRIPTION OF THE INVENTION

Now there is invented an improved method and equipment implementing the method, that enable detection of image frames from which a decoder can start the decoding process. Various aspects of the invention include methods, a video encoder, a video decoder, and computer programs that are characterized by what is stated in the independent claims. The preferred embodiments of the invention are disclosed in the dependent claims.

The invention is based on the idea of encoding a video sequence comprising an independent sequence of image frames, wherein at least one reference picture is predictable from at least one previous image frame that is earlier than the previous reference image frame in decoding order. An indication of at least one image frame is encoded into the video sequence, which indicated image frame is the first picture, in decoding order, of the independent sequence, said at least one reference image frame being included in the sequence. Respectively, in the decoding phase, the indication of at least one image frame is decoded from the video sequence, and the decoding of the video sequence is started from said first image frame of the independent sequence, whereby the video sequence is decoded without prediction from any image frame decoded prior to said first image frame.

Consequently, the idea of the present invention is to determine an initiation picture in an independently decodable group of pictures, whereby in the decoding phase any picture prior to said initiation picture is defined as unusable for reference. Accordingly, after decoding the initiation picture, all following pictures of the independently decodable sequence can be decoded without prediction from any picture decoded prior to said initiation picture.

According to an embodiment, the indication is encoded into the video sequence as a separate flag included in the header of a slice.

According to an embodiment, identifier values for pictures are encoded according to a numbering scheme, and the identifier value for the indicated first picture of an independent sequence is reset, preferably to zero.

According to an embodiment, an identifier value for the independent sequence is encoded into the video sequence.

An advantage of the procedure of the invention is that it enables to start browsing a video sequence from a random point, i.e. it provides the decoder with the information about the first picture of an independently decodable sequence. Thus, the decoder knows that by decoding this first picture, it is possible to continue the decoding process without any prediction from any prior picture. Accordingly, a further advantage is that the decoder may discard any picture decoded prior to said initiation picture from its buffer memory, since the prior pictures are no longer needed in the decoding process. A further advantage is that the procedure of the invention allows a separate video sequence to be easily inserted into another video sequence.

A further advantage is that enables identification of a picture boundary between two back-to-back initiation pictures by looking at the sub-sequence number of the initiation pictures. A yet further advantage relates to detection of losses of image frames that start an independently decodable sub-sequence. If such an image frame is lost e.g. during transmission, it is unlikely that any error concealment method would

5

result into a subjectively satisfactory image quality. It is therefore an advantage that decoders are provided means to detect losses of image frames that start an independently decodable sub-sequence. Decoders may react to such a loss by requesting a retransmission or picture refresh, for example.

BRIEF DESCRIPTION OF THE DRAWINGS

In the following, the invention will be described in connection with the preferred embodiments and with reference to the accompanying drawings, in which

FIG. 1 illustrates a common multimedia data streaming system in which the scalable coding hierarchy of the invention can be applied;

FIG. 2 illustrates a scalable coding hierarchy of a preferred embodiment of the invention;

FIGS. 3a and 3b illustrate embodiments of the invention for adjusting scalability;

FIGS. 4a, 4b and 4c illustrate embodiments of the invention for adjusting image numbering;

FIGS. 5a, 5b and 5c illustrate embodiments of the invention for using B-frames in a scalable coding hierarchy;

FIGS. 6a, 6b and 6c illustrate scalable coding hierarchies of preferred embodiments of the invention in connection with reference picture selection; and

FIG. 7 illustrates an arrangement according to a preferred embodiment of the invention for coding scene transition.

DETAILED DESCRIPTION OF THE INVENTION

In the following, a general-purpose multimedia data streaming system is disclosed, the basic principles of which can be applied in connection with any telecommunications system. Although the invention is described here with a particular reference to a streaming system, in which the multimedia data is transmitted most preferably through a telecommunications network employing a packet-switched data protocol, such as an IP network, the invention can equally well be implemented in circuit-switched networks, such as fixed telephone networks PSTN/ISDN (Public Switched Telephone Network/integrated Services Digital Network) or in mobile communications networks PLMN (Public Land Mobile Network). Further, the invention can be applied in the streaming of multimedia files in the form of both normal streaming and progressive downloading, and for implementing video calls, for example.

It is also to be noted that although the invention is described here with a particular reference to streaming systems and the invention can also be advantageously applied in them, the invention is not restricted to streaming systems alone, but can be applied in any video reproduction system, irrespective of how a video file that is to be decoded is downloaded and where it is downloaded from. The invention can therefore be applied for example in the playback of a video file to be downloaded from a DVD disc or from some other computer memory carrier, for example in connection with varying processing capacity available for video playback. In particular, the invention can be applied to different video codings of low bit rate that are typically used in telecommunications systems subject to bandwidth restrictions. One example is the system defined in the ITU-T standard H.263 and the one that is being defined in H.26L (possibly later to become H.264). In connection with these, the invention can be applied to mobile stations, for example, in which case the video playback can be made to adjust both to changing transfer capacity or channel

6

quality and to the processor power currently available, when the mobile station is used also for executing other applications than video playback.

It is further to be noted that, for the sake of clarity, the invention will be described below by giving an account of image frame coding and temporal predicting on image frame level. However, in practice coding and temporal predicting typically take place on block or macroblock level, as described above.

With reference to FIG. 1, a typical multimedia streaming system will be described, which is a preferred system for applying the procedure of the invention.

A multimedia data streaming system typically comprises one or more multimedia sources **100**, such as a video camera and a microphone, or video image or computer graphic files stored in a memory carrier. Raw data obtained from the different multimedia sources **100** is combined into a multimedia file in an encoder **102**, which can also be referred to as an editing unit. The raw data arriving from the one or more multimedia sources **100** is first captured using capturing means **104** included in the encoder **102**, which capturing means can be typically implemented as different interface cards, driver software, or application software controlling the function of a card. For example, video data may be captured using a video capture card and the associated software. The output of the capturing means **104** is typically either an uncompressed or slightly compressed data flow, for example uncompressed video frames of the YUV 4:2:0 format or motion-JPEG image format, when a video capture card is concerned.

An editor **106** links different media flows together to synchronize video and audio flows to be reproduced simultaneously as desired. The editor **106** may also edit each media flow, such as a video flow, by halving the frame rate or by reducing spatial resolution, for example. The separate, although synchronized, media flows are compressed in a compressor **108**, where each media flow is separately compressed using a compressor suitable for the media flow. For example, video frames of the YUV 4:2:0 format may be compressed using the low bit rate video coding according to the ITU-T recommendation H.263 or H.26L. The separate, synchronized and compressed media flows are typically interleaved in a multiplexer **110**, the output obtained from the encoder **102** being a single, uniform bit flow that comprises data of a plural number of media flows and that may be referred to as a multimedia file. It is to be noted that the forming of a multimedia file does not necessarily require the multiplexing of a plural number of media flows into a single file, but the streaming server may interleave the media flows just before transmitting them.

The multimedia files are transferred to a streaming server **112**, which is thus capable of carrying out the streaming either as real-time streaming or in the form of progressive downloading. In progressive downloading the multimedia files are first stored in the memory of the server **112** from where they may be retrieved for transmission as need arises. In real-time streaming the editor **102** transmits a continuous media flow of multimedia files to the streaming server **112**, and the server **112** forwards the flow directly to a client **114**. As a further option, real-time streaming may also be carried out such that the multimedia files are stored in a storage that is accessible from the server **112**, from where real-time streaming can be driven and a continuous media flow of multimedia files is started as need arises. In such case, the editor **102** does not necessarily control the streaming by any means. The streaming server **112** carries out traffic shaping of the multimedia data as regards the bandwidth available or the maximum

decoding and playback rate of the client **114**, the streaming server being able to adjust the bit rate of the media flow for example by leaving out B-frames from the transmission or by adjusting the number of the scalability layers. Further, the streaming server **112** may modify the header fields of a multiplexed media flow to reduce their size and encapsulate the multimedia data into data packets that are suitable for transmission in the telecommunications network employed. The client **114** may typically adjust, at least to some extent, the operation of the server **112** by using a suitable control protocol. The client **114** is capable of controlling the server **112** at least in such a way that a desired multimedia file can be selected for transmission to the client, in addition to which the client is typically capable of stopping and interrupting the transmission of a multimedia file.

When the client **114** is receiving a multimedia file, the file is first supplied to a demultiplexer **116**, which separates the media flows comprised by the multimedia file. The separate, compressed media flows are then supplied to a decompressor **118** where each separate media flow is decompressed by a decompressor suitable for each particular media flow. The decompressed and reconstructed media flows are supplied to a playback unit **120** where the media flows are rendered at a correct pace according to their synchronization data and supplied to presentation means **124**. The actual presentation means **124** may comprise for example a computer or mobile station display, and loudspeaker means. The client **114** also typically comprises a control unit **122** that the end user can typically control through a user interface and that controls both the operation of the server, through the above-described control protocol, and the operation of the playback unit **120**, on the basis of the instructions given by the end user.

It is to be noted that the transfer of multimedia files from the streaming server **112** to the client **114** takes place through a telecommunications network, the transfer path typically comprising a plural number of telecommunications network elements. It is therefore possible that there is at least some network element that can carry out traffic shaping of multimedia data with regard to the available bandwidth or the maximum decoding and playback rate of the client **114** at least partly in the same way as described above in connection with the streaming server.

Scalable coding will be described below with reference to a preferred embodiment of the invention and an example illustrated in FIG. 2. FIG. 2 shows part of a compressed video sequence having a first frame **200**, which is an INTRA frame, or I-frame, and thus an independently determined video frame the image information of which is determined without using motion-compensated temporal prediction. The I-frame **200** is placed on a first scalability layer, which may be referred to as an INTRA layer. Each scalability layer is assigned a unique identifier, such as the layer number. The INTRA layer may therefore be given the number 0, for example, or some other alphanumeric identifier, for example a letter, or a combination of a letter and a number.

Correspondingly, sub-sequences consisting of groups of one or more video frames are determined for each scalability layer, at least one of the images in a group (typically the first or the last one) being temporally predicted at least from a video frame of another, sub-sequence of typically either a higher or the same scalability layer, the rest of the video frames being temporally predicted either from only the video frames of the same sub-sequence, or possibly also from one or more video frames of said second sub-sequence. A sub-sequence may be decoded independently, irrespective of other sub-sequences, except said second sub-sequence. The sub-sequences of each scalability layer are assigned a unique

identifier using for example consecutive numbering starting with number 0 given for the first sub-sequence of a scalability layer. Since the I-frame **200** is determined independently and can also be decoded independently upon reception, irrespective of other image frames, it also forms in a way a separate sub-sequence.

An essential aspect of the present invention is therefore to determine each sub-sequence in terms of those sub-sequences the sub-sequence is dependent on. In other words, a sub-sequence comprises information about all the sub-sequences that have been directly used for predicting the image frames of the sub-sequence in question. This information is signalled in a video sequence bit stream, preferably separate from the actual image information, and therefore the image data of the video sequence can be preferably adjusted because it is easy to determine video sequence portions that are to be independently decoded and can be removed without affecting the decoding of the rest of the image data.

Next, within each sub-sequence, the video frames of the sub-sequence are given image numbers, using for example consecutive numbering that starts with the number 0 given to the first video frame of the sub-sequence. Since the I-frame **200** also forms a separate sub-sequence, its image number is 0. In FIG. 2, the I-frame **200** shows the type (I), sub-sequence identifier and image number (0.0) of the frame.

FIG. 2 further shows a next I-frame **202** of the INTRA layer, the frame thus being also an independently determined video frame that has been determined without using motion-compensated temporal prediction. The temporal transmission frequency of I-frames depends on many factors relating to video coding, image information contents and the bandwidth to be used, and, depending on the application or application environment, I-frames are transmitted in a video sequence at intervals of 0.5 to 10 seconds, for example. Since the I-frame **202** can be independently decoded, it also forms a separate sub-sequence. Since this is the second sub-sequence in the INTRA-layer, the consecutive numbering of the sub-sequence identifier of the I-frame **202** is 1. Further, since the I-frame **202** also forms a separate sub-sequence, i.e. it is the only video frame in the sub-sequence, its image number is 0. The I-frame **202** can thus be designated with identifier (I.1.0.) Correspondingly, the identifier of the next I-frame on the INTRA layer is (I.2.0.), etc. As a result, only independently determined I-frames in which the image information is not determined using motion-compensated temporal prediction are coded into the first scalability layer, i.e. the INTRA layer. The sub-sequences can also be determined using other kind of numbering or other identifiers, provided that the sub-sequences can be distinguished from one another.

The next scalability layer, which has a layer number 1, for example, and which may be referred to as the base layer, comprises coded, motion-compensated INTER or P-frames typically predicted only from previous image frames, i.e. in this case from the I-frames of an upper INTRA layer. The image information of the first P-frame **204** of the base layer shown in FIG. 2 is determined using the I-frame **200** of the INTRA layer. A P-frame **204** begins the first sub-sequence of the base layer, and therefore the sub-sequence identifier of the P-frame **204** is 0. Further, since the P-frame **204** is the first image frame of the first sub-sequence of the base layer, the image number of the P-frame **204** is 0. The P-frame **204** can thus be identified with (P.0.0).

The temporally succeeding P-frame **206** of the base layer is predicted from the previous P-frame **204**. The P-frames **204** and **206** thus belong to the same sub-sequence, whereby the P-frame **206** also receives the sub-sequence identifier 0. Since the P-frame **206** is the second image frame in the sub-se-

quence 0, the image number of the P-frame 206 is 1, and the P-frame 206 can be identified with (P.0.1).

The scalability layer succeeding the base layer and having the layer number 2, is called enhancement layer 1. This layer comprises coded, motion-compensated P-frames predicted only from previous image frames, in this case either from I-frames of the INTRA layer or P-frames of the base layer. FIG. 2 shows a first image frame 208 and a second image frame 210 of enhancement layer 1, which are both predicted only from the first image frame 200 of the INTRA layer. The P-frame 208 begins the first sub-sequence of enhancement layer 1, the sub-sequence identifier of the P-frame thus being 0. Further, since the P-frame 208 is the first and only image frame in said sub-sequence, the P-frame 208 receives the image number 0. The P-frame 208 can thus be identified with (P.0.0).

Since the second image frame 210 is also predicted only from the first image frame 200 of the INTRA layer, the P-frame 210 begins the second sub-sequence of enhancement layer 1 and the sub-sequence identifier of the P-frame 210 is therefore 1. Since the P-frame 210 is the first image frame in the sub-sequence, the image number of the P-frame 210 is 0. The P-frame can thus be identified with (P.1.0). The temporally succeeding P-frame 212 of enhancement layer 1 is predicted from the previous P-frame 210. The P-frames 210 and 212 thus belong to the same sub-sequence, and therefore the P-frame also receives the sub-sequence identifier 1. The P-frame 212 is the second image frame in the sub-sequence 1, and therefore the P-frame receives the image number 1 and can be identified with (P.1.1).

The temporally fourth image frame 214 of enhancement layer 1 is predicted from the first image frame 204 of the base layer. The P-frame 214 thus begins a third sub-sequence of enhancement layer 1 and therefore the P-frame 214 receives the sub-sequence identifier 2. Further, since the P-frame 214 is the first and only image frame in the sub-sequence, the image number of the P-frame 214 is 0. The P-frame 208 can therefore be identified with (P.2.0).

Also the temporally fifth image frame 216 of enhancement layer 1 is predicted only from the first image frame 204 of the base layer, the P-frame 216 thus beginning the fourth sub-sequence of enhancement layer 1, and the sub-sequence identifier of the P-frame 216 is 3. In addition, since the P-frame 216 is the first one in the sub-sequence in question, the image number of the P-frame 216 is 0. The P-frame 216 can therefore be identified with (P.3.0). The temporally following P-frame 218 of enhancement layer 1 is predicted from the previous P-frame 216. The P-frames 216 and 218 thus belong to the same sub-sequence, and the sub-sequence identifier of the P-frame 218 is also 3. Since the P-frame 218 is the second image frame in the sub-sequence 3, the image number of the P-frame 218 is 1 and the identifier of the P-frame 218 is (P.3.1).

For simplicity and clarity of illustration the above disclosure only relates to I- and P-frames. However, a person skilled in the art will find it apparent that the scalable video coding of the invention can also be implemented using other known image frame types, such as the above described B-frames and at least SI-frames, SP-frames and MH-frames. SI-frames correspond to I-frames, but together with an SP-frame they allow an identical image to be reconstructed. An SP-frame, in turn, is a P-frame subjected to a particular coding that allows an identical image to be reconstructed together with an SI-frame or another SP-frame. SP-frames are typically placed into a video sequence into points where an access point or a scanning point is desired, or where the changing of the coding parameters of the video stream should be possible. The

frames can also be used for error correction and for increasing error tolerance. SP-frames are otherwise similar to ordinary P-frames predicted from previous frames, except that they are defined so that they can be replaced by another video frame of the SP- or SI-type, the result of the decoding of the new frame being identical with the decoding result of the original SP-frame that was in the video stream. In other words, a new SP-frame that is used for replacing the one that was in the video stream is predicted from another sequence or video stream and yet the reconstructed frame has identical contents. SP-frames are described for example in the Applicant's earlier application PCT/FI02/00004.

Similarly as B-frames, macroblocks of MH (Multi Hypothesis) frames, based on motion-compensated prediction, are predicted from two other frames, which are not, however, necessarily located next to an MH-frame. More precisely, the predicted macroblocks are computed as an average of two macroblocks of two other frames. Instead of two frames, MH-frame macroblocks can naturally be also predicted from one other frame. Reference images may change according to macroblock, in other words, all macroblocks in one and the same image are not necessarily predicted using the same frames.

A sub-sequence thus covers a specific period of time in a video sequence. The sub-sequences of the same layer or of different layers may be partly or entirely overlapping. If there are temporally overlapping image frames on the same layer, the frames are interpreted as alternative presentations of the same image content and therefore any mode of image presentation can be used. On the other hand, if there are temporally overlapping image frames on different layers, they form different presentations of the same image content, and therefore presentations differ in image quality, i.e. the quality of image is better on a lower layer.

The above disclosure referring to FIG. 2 illustrates a scalable coding arrangement and a hierarchical structure and numbering of image frames according to a preferred embodiment of the invention. In this embodiment the INTRA-layer only comprises I-frames and the base layer can only be decoded using the information received from the INTRA-layer. Correspondingly, the decoding of enhancement layer 1 typically requires information from both the base layer and the INTRA-layer.

The number of the scalability layers is not restricted to three, as above, but any number of enhancement layers that is considered necessary for producing sufficient scalability may be used. Consequently, the layer number of enhancement layer 2 is four, that of enhancement layer 3 is five, etc. Since some of the image frames in the above example are given the same identifier (e.g. the identifier of both image frames 204 and 208 is (P.0.0)), by including the layer number in the identifier each image frame can be uniquely identified and, at the same time, the dependencies of each image frame on other image frames is preferably determined. Each image frame is thus uniquely identified, the identifier of image frame 204, for example, being (P.1.0.0), or simply (1.0.0) and, correspondingly, that of image 208 being (P.2.0.0), or (2.0.0).

According to a preferred embodiment of the invention, the number of a reference image frame is determined according to a specific, pre-determined alpha-numeric series, for example as an integer between 0 and 255. When the parameter value achieves the maximum value N (e.g. 255) in the series concerned, the determining of the parameter value starts from the beginning, i.e. from the minimum value of the series (e.g. 0). An image frame is thus uniquely identified within a specific sub-sequence up to the point where the same image number is used again. The sub-sequence identifier can

also be determined according to a specific, predetermined arithmetic series. When the value of the sub-sequence identifier achieves the maximum value N of the series, the determining of the identifier starts again from the beginning of the series. However, a sub-sequence cannot be assigned an identifier that is still in use (within the same layer). The series in use may also be determined in another way than arithmetically. One alternative is to assign random sub-sequence identifiers, taking into account that an assigned identifier is not be used again.

A problem in the numbering of image frames arises when the user wishes to start browsing a video file in the middle of a video sequence. Such situations occur for example when the user wishes to browse a locally stored video file backward or forward or to browse a streaming file at a particular point; when the user initiates the playback of a streaming file from a random point; or when a video file that is to be reproduced is detected to contain an error that interrupts the playback or requires the playback to be resumed from a point following the error. When the browsing of a video file is resumed from a random point after previous browsing, discontinuity typically occurs in the image numbering. The decoder typically interprets this as unintentional loss of image frames and unnecessarily tries to reconstruct the image frames suspected as lost.

According to a preferred embodiment of the invention, this can be avoided in the decoder by defining an initiation image in an independently decodable Group of Pictures GOP that is activated at the random point of the video file, and the number of the initiation image is set at zero. This independently decodable image group can thus be a sub-sequence of an INTRA-layer, for example, in which case an I-frame is used as the initiation image, or, if scaling originating from the base layer is employed, the independently decodable image group is a sub-sequence of the base layer, in which case the first image frame of the sub-sequence, typically an I-frame, is usually used as the initiation image. Consequently, when activated at a random point, the decoder preferably sets the identifier of the first image frame, preferably an I-frame, of the independently decodable sub-sequence at zero. Since the sub-sequence to be decoded may also comprise other image frames whose identifier is zero (for example when the above described alpha-numeric series starts from the beginning), the beginning of the sub-sequence, i.e. its first image frame, can be indicated to the decoder for example by a separate flag added to the header field of a slice of the image frame. This allows the decoder to interpret the image numbers correctly and to find the correct image frame that initiates the sub-sequence from the video sequence image frames.

The above numbering system provides only one example of how the unique image frame identification of the invention can be carried out so that interdependencies between the image frames are indicated at the same time. However, video coding methods in which the method of the invention can be applied, such as video coding methods according to the ITU-T standards H.263 and H.26L, employ code tables, which in turn use variable length codes. When variable length codes are used for coding layer numbers, for example, a lower code word index, i.e. a smaller layer number, signifies a shorter code word. In practice the scalable coding of the invention will be used in most cases in such a way that the base layer will consist significantly more image frames than the INTRA-layer. This justifies the use of a lower index, i.e. a smaller layer number, on the base layer than on the INTRA-layer, because the amount of coded video data is thereby advantageously reduced. Consequently, the INTRA-layer is preferably assigned layer number 1 and the base layer is given

layer number 0. Alternatively, the code can be formed by using fewer bits for coding the base layer number than the INTRA-layer number, in which case the actual layer number value is not relevant in view of the length of the code created.

Further, according to a second preferred embodiment of the invention, when the number of the scalability layers is to be kept low, the first scalability layer in particular can be coded to comprise both the INTRA-layer and the base layer. From the point of view of coding hierarchy, the simplest way to conceive this is to leave out the INTRA-layer altogether, and to provide the base layer with coded frames consisting of both independently defined I-frames, the image information of which has not been determined using motion-compensated temporal prediction, and image frames predicted from previous frames, which image frames in this case are motion-compensated P-frames predicted from the I-frames of the same layer. The layer number 0 can thus still be used for the base layer and, if enhancement layers are coded into the video sequence, enhancement layer 1 is assigned layer number 1. This is illustrated in the following, with reference to FIGS. 3a and 3b.

FIG. 3a shows a non-scalable video sequence structure, in which all image frames are placed on the same scalability layer, i.e. the base layer. The video sequence comprises a first image frame 300 which is an I-frame (I.0.0) and which thus initiates a first sub-sequence. The image frame 300 is used for predicting a second image frame 302 of the sub-sequence, i.e. a P-frame (P.0.1), which is then used for predicting a third image frame 304 of the sub-sequence, i.e. a P-frame (P.0.2), which is in turn used for predicting the next image frame 306, i.e. a P-frame (P.0.3). The video sequence is then provided with an I-frame (I.1.0) coded therein, i.e. an I-frame 308, which thus initiates a second sub-sequence in the video sequence. This kind of non-scalable coding can be used for example when the application employed does not allow scalable coding to be used, or there is no need for it. In a circuit-switched videophone application, for example, channel bandwidth remains constant and the video sequence is coded in real-time, and therefore there is typically no need for scalable coding.

FIG. 3b, in turn, illustrates an example of how scalability can be added, when necessary, to a combined INTRA- and base layer. Here, too, the video sequence base layer comprises a first image frame 310 which is an I-frame (I.0.0) and which initiates a first sub-sequence of the base layer. The image frame 310 is used for predicting a second image frame 312 of the sub-sequence, i.e. a P-frame (P.0.1), which is then used for predicting a third image frame 314 of the sub-sequence, i.e. a P-frame (P.0.2). Enhancement layer 1, however, is also coded into this video sequence and it comprises a first sub-sequence, the first and only image frame 316 of which is a P-frame (P.0.0), which is predicted from the first image frame 310 of the base layer. The first image frame 318 of a second sub-sequence of the enhancement layer is, in turn, predicted from the second image frame 312 of the base layer, and therefore the identifier of this P-frame is (P.1.0). The next image frame 320 of the enhancement layer is again predicted from the previous image frame 318 of the same layer and, therefore, it belongs to the same sub-sequence, its identifier thus being (P.1.1).

In this embodiment of the invention the sub-sequences of the base layer can be decoded independently, although a base layer sub-sequence may be dependent on another base layer sub-sequence. The decoding of the base layer sub-sequences requires information from the base layer and/or from the second sub-sequence of enhancement layer 1, the decoding of the sub-sequences of enhancement layer 2 requires informa-

13

tion from enhancement layer 1 and/or from the second sub-sequence of enhancement layer 2, etc. According to an embodiment, I-frames are not restricted to the base layer alone, but lower enhancement layers may also comprise I-frames.

The basic idea behind the above embodiments is that a sub-sequence comprises information about all the sub-sequences it is dependent on, i.e. about all sub-sequences that have been used for predicting at least one of the image frames of the sub-sequence in question. However, according to an embodiment it is also possible that a sub-sequence comprises information about all sub-sequences that are dependent on the sub-sequence in question, in other words, about all the sub-sequences in which at least one image frame has been predicted using at least one image frame of the sub-sequence in question. Since in the latter case the dependencies are typically determined temporally forward, image frame buffers can be advantageously utilized in the coding in a manner to be described later.

In all the above embodiments the numbering of the image frames is sub-sequence-specific, i.e. a new sub-sequence always starts the numbering from the beginning. The identification of an individual image frame thus requires the layer number, sub-sequence identifier and image frame number to be determined. According to a preferred embodiment of the invention, the image frames can be independently numbered using consecutive numbering in which successive reference image frames in the coding order are indicated with numbers incremented by one. As regards layer numbers and sub-sequence identifiers, the above-described numbering procedure can be used. This allows each image frame to be uniquely identified, when necessary, without using the layer number and sub-sequence identifier.

This is illustrated with the example shown in FIG. 4a in which the base layer comprises a temporally first I-frame 400 (I.0.0). This frame is used for predicting a first image frame 402 of enhancement layer 1, i.e. (P.0.1), which is then used for predicting a second image frame 404 belonging to the same sub-sequence (with sub-sequence identifier 0), i.e. (P.0.2), which is used for predicting a third image frame 406 of the same sub-sequence, i.e. (P.0.3), which is used for predicting a fourth image frame 408 (P.0.4) and, finally, the fourth frame for predicting a fifth image frame 410 (P.0.5). The temporally next video sequence image frame 412 is located on the base layer, where it is in the same sub-sequence as the I-frame 400, although temporally it is only the seventh coded image frame, and therefore its identifier is (P.0.6). The seventh frame is then used for predicting a first image frame 414 of the second sub-sequence of enhancement layer 1, i.e. (P.1.7), which is then used for predicting a second image frame 416 belonging to the same sub-sequence (with sub-sequence identifier 1), i.e. (P.1.8), which in turn used for predicting a third image frame 418 (P.1.9), the third for predicting a fourth image frame 420 (P.1.10) and, finally, the fourth for predicting a fifth image frame 422 (P.1.11) of the same sub-sequence. Again, the temporally next video sequence image frame 424 is located on the base layer, where it is in the same sub-sequence as the I-frame 400 and the P-frame 412, although temporally it is only the thirteenth coded image frame and therefore its identifier is (P.0.12). For clarity of illustration, the above description of the embodiment does not comprise layer identifiers, but it is apparent that in order to implement scalability, also the layer identifier must be signalled together with the video sequence, typically as part of the image frame identifiers.

FIGS. 4b and 4c show alternative embodiments for grouping the image frames of the video sequence shown in FIG. 4a.

14

The image frames in FIG. 4b are numbered according to sub-sequence, i.e. a new sub-sequence always starts the numbering from the beginning (from zero). FIG. 4c, in turn, employs image frame numbering which corresponds otherwise to that used in FIG. 4a, except that the P-frames of the base layer are replaced by SP-frame pairs to allow for identical reconstruction of image information.

As stated above, the procedure of the invention can also be implemented using B-frames. One example of this is illustrated in FIGS. 5a, 5b and 5c. FIG. 5a shows a video sequence in the time domain, the sequence comprising P-frames P1, P4 and P7, with B-frames placed between them, the interdependencies of the B-frames with regard to temporal predicting being shown with arrows. FIG. 5b shows a preferred grouping of video sequence image frames in which the interdependencies shown in FIG. 5a are indicated. FIG. 5b illustrates sub-sequence-specific image frame numbering in which a new sub-sequence always starts the numbering of the image frames from zero. FIG. 5c, in turn, illustrates image frame numbering, which is consecutive in the order of temporal prediction, wherein the following reference frame always receives the next image number as the previously encoded reference frame. The image frame (B1.8) (and (B2.10)) does not serve as a reference prediction frame to any other frame, therefore it does not affect the image frame numbering.

The above examples illustrate different alternatives of how scalability of video sequence coding can be adjusted by using the method of the invention. From the point of view of the terminal device reproducing the video sequence, the more scalability layers are available, or the more scalability layers it is capable of decoding, the better the image quality. In other words, increase in the amount of image information and in the bit rate used for transferring the information improves the temporal or spatial resolution, or the spatial quality of the image data. Correspondingly, a higher number of scalability layers also sets considerably higher demands on the processing capacity of the terminal device performing decoding.

In addition, the above examples illustrate the advantage gained by using sub-sequences. With image frame identifiers, the dependencies of each image frame from other image frames in the sub-sequence are indicated in an unambiguous manner. A sub-sequence thus forms an independent whole that can be left out of a video sequence, when necessary, without affecting the decoding of subsequent image frames of the video sequence. In that case, only the image frames of the sub-sequence in question and of those sub-sequences on the same and/or lower scalability layers dependent on it are not decoded.

The image frame identifier data transmitted together with the video sequence are preferably included in the video sequence header fields or in the header fields of the transfer protocol to be used for transferring the video sequence. In other words, the identifier data of predicted image frames are not included in the image data of the coded video sequence, but always into the header fields, whereby the dependencies of the image frames can be detected without decoding the images of the actual video sequence. The identifier data of the image frames can be stored for example in the buffer memory of the streaming server as the video sequence is being coded for transmission. In addition, the sub-sequences can be independently decoded on each scalability layer, because the image frames of a sub-sequence are not dependent on other sub-sequences of the same scalability layer.

According to an embodiment of the invention, the image frames comprised by a sub-sequence may thus depend also on other sub-sequences of the same scalability layer. This dependency must then be signalled for example to the streaming

server carrying out traffic shaping, because interdependent sub-sequences located on the same layer cannot be separately removed from a video sequence to be transmitted. A preferred way to carry out the signalling is to include it in the image frame identifiers to be transmitted, for example by listing the layer-sub-sequence pairs the sub-sequence in question depends on. This also provides a preferred way of indicating dependency from another sub-sequence of the same scalability layer.

The above examples illustrate a situation where image frames are temporally predicted from previous image frames. In some coding methods, however, the reference picture selection has been further extended to also include the predicting of the image information of image frames from temporally succeeding image frames. Reference picture selection offers most diversified means for creating different temporally scalable image frame structures and allows the error sensitivity of the video sequence to be reduced. One of the coding techniques based on reference picture selection is INTRA-frame postponement. The INTRA-frame is not placed into its temporally "correct" position in the video sequence, but its position is temporally postponed. The video sequence image frames that are between the "correct" position of the INTRA-frame and its actual position are predicted temporally backward from the INTRA-frame in question. This naturally requires that uncoded image frames be buffered for a sufficiently long period of time so that all image frames that are to be displayed can be coded and arranged into their order of presentation. INTRA-frame transfer and the associated determining of sub-sequences in accordance with the invention are illustrated in the following with reference to FIG. 6.

FIG. 6a shows a video sequence part in which the INTRA-frame comprises a single I-frame 600, which is temporally transferred to the position shown in FIG. 6, although the "correct" position of the I-frame in the video sequence would have been at the first image frame. The video sequence image frames between the "correct" position and the real position 600 are thus temporally predicted backward from the I-frame 600. This is illustrated by a sub-sequence coded into enhancement layer 1 and having a first temporally backward predicted image frame 602, which is a P-frame (P.0.0). This frame is used for temporally predicting a previous image frame 604, i.e. a P-frame (P.0.1), which is used in turn for predicting an image frame 606, i.e. a P-frame (P.0.2), and, finally, the frame 606 for predicting an image frame 608, i.e. a P-frame (P.0.3), which is at the position that would have been the "correct" position of the I-frame 600 in the video sequence. Correspondingly, the I-frame 600 on the base layer is used for temporally forward prediction of a sub-sequence comprising four P-frames 610, 612, 614 and 616, i.e. P-frames (P.0.0), (P.0.1), (P.0.2) and (P.0.3).

The fact that in this example backward predicted image frames are placed on a lower layer than forward predicted image layers indicates that for purposes of illustration, backward predicted image frames are in this coding example considered subjectively less valuable than forward predicted image frames. Naturally the sub-sequences could both be placed on the same layer, in which case they would be considered equal, or a backward predicted sub-sequence could be on the upper layer, in which case it would be considered subjectively more valuable.

FIGS. 6b and 6c show some alternatives for coding a video sequence according to FIG. 6a. In FIG. 6b both forward and backward predicted sub-sequences are placed on the base layer, the I-frame being only located on the INTRA-layer. The forward predicted sub-sequence on this layer is thus the

second sub-sequence and its sub-sequence identifier is 1. In FIG. 6c, in turn, an I-frame and a forward predicted sub-sequence based on it are located on the base layer, while a backward predicted sub-sequence is located on enhancement layer 1.

Moreover, according to a preferred embodiment of the invention, the above-described scalability can be utilized for coding what is known as a scene transition into a video sequence. Video material, such as news reports, music videos and movie trailers, often comprise rapid cuts between separate image material scenes. Sometimes the cuts are abrupt, but often a procedure known as scene transition is used in which transfer from one scene to another takes place by dimming, wiping, mosaic dissolving or scrolling the image frames of a previous scene, and, correspondingly, by presenting those of a later scene. From the point of view of coding efficiency, the video coding of a scene transition is often most problematic, because the image frames appearing during the scene transition comprise information on the image frames of both the terminating and the initiating scene.

A typical scene transition, fading, is carried out by gradually reducing the intensity or luminance of the image frames of a first scene to zero, while gradually increasing the intensity of the image frames of a second scene to its maximum value. This scene transition is referred to as cross-faded scene transition.

Generally speaking, a computer-made image can be thought of as consisting of layers, or image objects. Each object can be defined with reference to at least three information types: the structure of the image object, its shape and transparency, and the layering order (depth) in relation to the background of the image and to other image objects. Shape and transparency are often determined using what is known as an alpha plane, which measures opacity and the value of which is usually determined separately for each image object, possibly excluding the background, which is usually determined as non-transparent. The alpha plane value of a non-transparent image object, such as the background, can thus be set at 1.0, whereas the alpha plane value of a fully transparent image object is 0.0. The values in between define the intensity of the visibility of a specific image object in a picture in proportion to the background and to other, at least partly overlapping, image objects that have a higher depth value than the image object in question.

The superimposition of image objects in layers according to their shape, transparency and depth position is referred to as scene composition. In practice the procedure is based on the use of weighted averages. First, the image object that is closest to the background, i.e. deepest according to its depth position, is placed onto the background and a combined image is formed of the two. The pixel values of the combined image are formed as an average weighted by the alpha plane values of the background image and the image object in question. The alpha plane value of the combined image is then set at 1.0, after which it serves as a background image for the next image object. The process continues until all image objects are attached to the image.

In the following, a procedure according to a preferred embodiment of the invention will be described in which video sequence scalability layers are combined with the above described image objects of image frames and their information types to provide a scene transition with scalable video coding that also has good compression efficiency.

This embodiment of the invention is illustrated in the following by way of example and in a simplified manner by using cross-faded scene transition, on one hand, and abrupt scene transition, on the other hand, as examples. The image

17

frames to be displayed during a scene transition are typically formed of two superimposed image frames, a first image frame comprising a first image scene and a second image frame a second scene. One of the image frames serves as the background image and other, which is referred to as a foreground image, is placed on top of the background image. The opacity of the background image, i.e. its non-transparency value, is constant. In other words, its pixel-specific alpha plane values are not adjusted.

In this embodiment of the invention, the background and foreground images are both defined according to scalability layer. This is illustrated in FIG. 7, which shows an example of how image frames of two different scenes can be placed on scalability layers during a scene transition of the invention. FIG. 7 shows a first image frame 700 of a first (terminating) scene positioned on the base layer. The image frame 700 may be either an I-frame containing image information that has not been determined using motion-compensated temporal predicting, or it may be a P-frame that is a motion-compensated image frame predicted from previous image frames. The coding of a second (initiating) scene starts during the temporally following image frame and, according to the invention, the image frames of the scene are also placed on the base layer. Remaining image frames 702, 704 of the second (terminating) scene are then placed on enhancement layer 1. These image frames are typically P-frames.

In this embodiment, the image frames of the second (initiating) scene are thus placed on the base layer, at least for the duration of the scene transition. The first image frame 706 of the scene is typically an I-frame, and it is used for temporally predicting the succeeding image frames of the scene. Consequently, the succeeding image frames of the second scene are temporally predicted frames, typically P-frames, such as frames 708 and 710 shown in FIG. 7.

According to a preferred embodiment of the invention, this placing of image frames on scalability layers can be used for implementing a cross-faded scene transition by determining the image layer that is on the base layer always as a background image of maximum opacity (100%), or non-transparency value. During a scene transition, image frames located on enhancement layers are placed onto the background image and their opacity is adjusted for example by means of suitable filters such that the frames gradually change from non-transparent to transparent.

In the video sequence of FIG. 7, there are no image frames on the lower scalability layers during the first base layer image frame 700. For this time instant, the first image frame 700 is only coded into the video sequence.

The next image frame 706 of the base layer initiates a new (second) scene, during which the image frame 706 is provided with depth positioning that places it as the background image, and its opacity value is set to the maximum. Temporally simultaneously with the image frame 706 of the base layer, there is an image frame 702 of a terminating (first) scene on enhancement layer 1. To allow a cross-faded scene transition to be produced, the transparency of the frame 702 must be increased. The example of FIG. 7 assumes that the opacity of the image frame 702 is set at 67% and, in addition, the image frame 702 is provided with depth positioning that determines it as a foreground image. For this time instant, an image combining the image frames 706 and 702 is coded into the video sequence, image 706 being visible as a weaker image on the background and image 702 as a stronger image at the front, because its opacity value is essentially high (67%).

During the temporally following image frame, there is a second image frame 708 of the second scene on the base layer,

18

the frame 708 being thus correspondingly provided with depth positioning determining it as a background image, and its opacity value is set to the maximum. Enhancement layer 1 further comprises the last image frame 704 of a temporally simultaneously terminating (first) scene, the opacity value of the frame being set at 33% and, in addition, the image frame 704 being provided with depth positioning that determines it as a foreground image as well. Consequently, for this time instant, an image combined of the image frames 708 and 704 is coded into the video sequence, the image 708 being displayed as a stronger image on the background and the image 704 as a weaker image on the foreground, because the opacity value of the image 704 is no longer more than 33%.

During the temporally following image frame, the base layer comprises a third image frame 710 of the second scene. Since the first scene has terminated, only the image frame 710 is coded into the video sequence, and the displaying of the second scene continues from the frame 710.

The above disclosure describes, by way of example, the positioning of image frames according to the invention on scalability layers to implement cross-faded scene transition in a manner that is advantageous from the point of view of coding efficiency. However, it is possible that when a video sequence is being transmitted or decoded, a situation arises in which the bit rate of the video sequence must be adjusted according to the maximum value of the bandwidth and/or terminal device decoding rate available for data transfer. This kind of bit rate control causes problems when the scene transition is to be implemented using prior art video coding methods.

A preferred embodiment of the present invention now allows one or more scalability layers, or independently decodable sub-sequences included in them, to be removed from a video sequence, whereby the bit rate of the video sequence can be decreased and yet, at the same time, the video sequence can be decoded without reducing image frequency. In the image frame positioning according to FIG. 7, this can be implemented by removing enhancement layer 1 from the video sequence. The video sequence is thus only used for displaying the image frames 700, 706, 708 and 710 of the base layer. In other words, a direct transition from the first (terminating) scene to the second (initiating) scene takes place in the form of an abrupt scene transition, i.e. directly from the image frame 700 of the first scene into the I-image frame 706 that initiates the second scene. The transition is thus not a cross-faded scene transition but an abrupt scene transition. Nevertheless, the scene transition can be carried out in an advantageous manner without affecting the quality of the video sequence image, and the viewer usually does not experience an abrupt scene transition carried out instead of a cross-faded scene transition in any way disturbing or faulty. On the contrary, since the prior art implementation does not allow scalability layers to be removed, scene transition would often require image frequency to be reduced, which the viewer would find jerky and disturbing.

The invention thus provides a preferred means for carrying out multimedia data traffic shaping in a streaming server comprising information about the different sub-sequences of a video sequence: their average bit rate, location in relation to the entire video sequence, duration and their interdependencies regarding the layers. The streaming server also determines the maximum value of the bandwidth available for the data transfer and/or the decoding rate of the terminal device. On the basis of this information, the streaming server decides how many scalability layers and which sub-sequences are transmitted in the video sequence. Bit rate control can thus be carried out, when necessary, by making first a rough adjust-

ment of the number of the scalability layers, after which finer sub-sequence-specific adjustment can be easily carried out. At its simplest, bit rate control means making sub-sequence-specific decisions on whether a particular sub-sequence will be added to a video sequence or removed from it. In case of removal it is advisable to remove entire sub-sequences from a video sequence, because the removal of separate images may cause errors in other images of the same sub-sequence. For the same reason, all sub-sequences of a lower enhancement layer should be left out if they are dependent on the removed sub-sequence of a higher layer. If there are interdependent sub-sequences on one and the same scalability layer, sub-sequences dependent on an earlier sub-sequence must be removed if the earlier sub-sequence is removed.

If the image frame identifier data are added to a video sequence that is to be transmitted, traffic shaping can also be carried out in a telecommunications network element to be used for the transfer of the video sequence, for example in an Internet router, in different gateways, or at a base station or base station controller of a mobile communications network. For the network element to be able to maintain and process the sub-sequence information, it must have extra memory and processing capacity. For this reason traffic shaping that is to be carried out in the network is perhaps most probably executed using simple processing methods, such as the Diff-Serv, i.e. differentiated services, procedure that is supported by some IP-based networks. In the DiffServ method, each IP data packet is assigned a priority, whereby data packets of a higher priority are relayed faster and more reliably to the recipient than packets of a lower priority. This is advantageously applied to the scalability of the invention by determining not only scalability-layer-specific, but also sub-sequence-specific priorities, which enables a highly advanced prioritisation.

There are many alternatives for adding image frame identifier data to a video sequence that is to be transmitted. In addition, it is also possible not to include any identifier data into the video sequence, in which case traffic shaping is only carried out at the streaming server. The identifier data can be included in the header fields of a video sequence, or in the header fields of the transfer protocol to be used, such as RTP (Real Time Protocol). According to a preferred embodiment, the identifier data can be transferred using a Supplemental Enhancement Information (SEI) mechanism. SEI provides a data delivery mechanism that is transferred synchronously with the video data content, thus assisting in the decoding and displaying of the video sequence. The SEI mechanism, particularly when used for transferring layer and sub-sequence information, is disclosed more in detail in the ITU-T standard document ITU-T Rec. H.264 (ISO/IEC 14496-10:2002), Annex D. In the cases, wherein a separate transfer protocol or mechanism is used for identifier data transfer, traffic shaping can be carried out also at one of the network elements of the transfer path. In addition, the receiving terminal device can control the decoding.

If the encoder or decoder supports reference picture selection, video sequence coding requires that decoded image frames be buffered before the coding so as to allow the relationships between different image frames to be temporally predicted from one or more other image frames. Image frame buffering can be arranged at least in two different ways, either as sliding windowing or as adaptive buffer memory control. In sliding windowing, M image frames that were coded last are used as a buffer. The frames in the buffer memory are in a decoded and reconstructed form, which allows them to be used as reference images in the coding. As the coding proceeds, the image frame buffering functions on the basis of the

FIFO principle (First-In-First-Out). Images that are not used as reference images, such as conventional B-images, do not need to be stored in the buffer. Alternatively, the buffering can be also be implemented as adaptive buffer memory control, in which case the image buffering is not restricted to the FIFO principle, but image frames that are not needed can be removed from the buffer in the middle of the process, or, correspondingly, some image frames can be stored in the buffer for a longer period of time, if they are needed as reference images for later image frames. A known reference picture selection is implemented by indexing image frames that are in the buffer memory into a specific order, the image indices being then used to refer to an image in connection with motion-compensation, for example. This indexing method generally provides better compression efficiency compared to using image numbers, for example, for referring to a specific image when motion-compensation reference images are to be signalled.

The above reference image indexing method is sensitive to transfer errors, because the buffers of the sender's encoder and the recipient's decoder must contain mutually corresponding reconstructed images in identical order to ensure that the encoder and decoder both form the same indexing order. If the image frames are indexed in different order in the buffers of the encoder and the decoder, an incorrect reference image may be used in the decoder. To prevent this, it is essential that the decoder can be controlled to take into account image frames and sub-sequences that the encoder has intentionally removed from the video sequence. In that case the image frame numbering may comprise gaps, which the decoder typically interprets as errors and tries to reconstruct the image frames interpreted as lost. For this reason, it is essential that the encoder is capable to inform the decoder that the discontinuities in the image numbering of the transmitted image frames are intentional.

In response to this, and provided that sliding windowing is used for buffering the image frames, the decoder enters into the buffer memory a number of image frames, the contents of which may be fully random, corresponding to the missing image numbers. These random image frames are then designated by an identifier "invalid" to indicate that the frames in question do not belong to the actual video sequence, but are only filler frames entered for purposes of buffer memory management. A filler frame can naturally be implemented using only memory indicators, i.e. no data is preferably entered into the buffer memory, but memory management is used merely to store a reference to a generic "invalid" frame. The entering of the image frames of the actual video sequence continues from the correct image frame number after the number of filler frames indicated by the missing image numbers has been entered into the buffer, which allows the buffer memories of the encoder and the decoder to be kept preferably in synchronism. If during decoding a reference to an image number is detected which is then found to indicate a filler frame located in the buffer, error correction actions are initiated in the decoder to reconstruct the actual reference image, for example by asking the encoder to re-transmit the reference image in question.

Further, the procedure of the invention allows separate buffer memories to be used on the different scalability layers, or, correspondingly, sub-sequence-specifically. Each scalability layer may thus have a separate buffer memory that is conceptually separate and functions on the basis of the sliding window principle. Similarly, each sub-sequence may also be provided with a conceptually separate buffer memory that also functions on the basis of the sliding window principle. This means that the buffer memory is always emptied when a

sub-sequence terminates. Separate buffer memories can be used in a preferred manner for reducing the need for signaling in certain situations in which ordinary sliding window buffering would be inadequate and actively adaptive buffer memory management would need to be used instead.

The H.26L standard defines a picture order count as a picture position in output order. The decoding process specified in the H.26L standard uses picture order counts to determine default index orderings for reference pictures in B slices, to represent picture order differences between frames and fields for vector scaling in motion vector prediction and for implicit mode weighted prediction in B slices, and to determine when successive slices in decoding order belong to different pictures. The picture order count is coded and transmitted for each picture.

In one embodiment of the invention, the decoder uses the picture order count to conclude that pictures are temporally overlapping, i.e., pictures that have an equal picture order count are temporally overlapping. Preferably, the decoder outputs only the picture on the highest received layer. In the absence of layer information, the decoder concludes that the latest temporally overlapping picture in decoding order resides on highest received layer.

The above disclosure describes a procedure for coding video frames for the purpose of producing a scalable, compressed video sequence. The actual procedure is carried out in a video encoder, such as the compressor **108** of FIG. 1, which may be any known video encoder. For example a video encoder according to the ITU-T recommendation H.263 or H.26L may be used, the video encoder being arranged to form, in accordance with the invention, a first sub-sequence into a video sequence, at least part of the sub-sequence being formed by coding I-frames; to form at least a second sub-sequence into the video sequence, at least part of the sub-sequence being formed by coding at least P- or B-frames, and at least one video frame of the second sub-sequence being predicted from at least one video frame of the first sub-sequence; and to determine into the video sequence the identification data of at least the video frames of the second sub-sequence.

According to the procedure of the invention, each sub-sequence of a particular scalability layer is preferably independently decodable, naturally taking into account dependencies from higher scalability layers and possibly other sub-sequences of the same scalability layer. A scalably compressed video sequence such as the one described above can thus be decoded by decoding a first sub-sequence of a video sequence, at least part of the sub-sequence having been formed by coding at least I-frames, and by decoding at least a second sub-sequence of the video sequence, at least part of the second sub-sequence having been formed by coding at least P- or B-frames, and at least one video frame of the second sub-sequence having been predicted from at least one video frame of the first sub-sequence, and by determining the identification and dependency data of at least the video frames comprised by the second sub-sequence of the video sequence, and by reconstructing at least part of the video sequence on the basis of the sub-sequence dependencies.

The actual decoding takes places in the video decoder, such as the decompressor **118** of FIG. 1, which may be any known video decoder. For example, a low bit rate video decoder according to the ITU-T recommendation H.263 or H.26L may be used, which in this invention is arranged to decode a first sub-sequence of a video sequence, at least part of the sub-sequence having been formed by coding I-frames; to decode at least a second sub-sequence of the video sequence, at least part of the second sub-sequence having been formed

by coding at least P- or B-frames and at least one video frame of the second sub-sequence having been predicted from at least one video frame of the first sub-sequence. The video decoder is arranged to determine the identification and dependency data of at least the video frames comprised by the second sub-sequence of the video sequence and to reconstruct at least part of the video sequence on the basis of the dependencies of the sub-sequences.

An essential aspect in the operation of the streaming system of the invention is that the encoder and decoder are positioned at least so that the encoder is operationally connected to the streaming server and the decoder is operationally connected to the receiving terminal device. However, the different elements of the streaming system, terminal devices in particular, may include functionalities that allow two-way transfer of multimedia files, i.e. transmission and reception. The encoder and decoder can thus be implemented in the form of what is known as a video codec integrating both encoder and decoder functionalities.

It is to be noted that according to the invention the functional elements of the above described streaming system and its elements, such as the streaming server, video encoder, video decoder and terminal are preferably implemented by hardware solutions or as a combination of hardware and software. The coding and decoding methods of the invention are particularly suitable for implementation as computer software comprising computer-readable commands for executing the process steps of the invention. A preferred way of implementing the encoder and the decoder is to store them in a storage means as a program code that can be executed by a computer-like device, for example a personal computer (PC) or a mobile station, to provide coding/decoding functionalities on the device in question.

Another alternative is to implement the invention as a video signal comprising a scalably compressed video sequence which in turn comprises video frames coded according to at least a first and a second frame format, the video frames according to the first frame format being independent of other video frames, and the video frames of the second frame format being predicted from at least one of the other video frames. According to the invention, the video signal in question comprises at least a first sub-sequence, at least part of which has been formed by coding at least video frames of the first frame format; at least a second sub-sequence, at least part of which has been formed by coding at least video frames of the second frame format; and at least one video frame of the second sub-sequence having been predicted from at least one video frame of the first sub-sequence; and at least one data field that determines video frames belonging to the second sub-sequence.

It is apparent to a person skilled in the art that as technology advances the basic idea of the invention can be implemented in various ways. The invention and its embodiments are therefore not restricted to the above examples, but they may vary within the scope of the claims.

The invention claimed is:

1. A method for encoding a video sequence comprising an independent sequence of image frames, wherein all motion-compensated temporal prediction references of the independent sequence refer only to image frames within said independent sequence, the method comprising:

encoding into the video sequence an indication of at least one image frame, which is the first image frame, in decoding order, of the independent sequence;
encoding identifier values for the image frames according to a numbering scheme; and

23

resetting the identifier value for the indicated first image frame of the independent sequence.

2. A method according to claim 1, further comprising: encoding the indication into the video sequence as a separate flag included in the header of a slice.

3. A method according to claim 1, further comprising: encoding into the video sequence an identifier value for the independent sequence.

4. A video encoder comprising at least one processor and at least one memory including computer program code, the at least one memory and the computer program code configured to, with the at least one processor, cause the video encoder to encode a video sequence comprising an independent sequence of image frames, wherein all motion-compensated temporal prediction references of the independent sequence refer only to image frames within said independent sequence by causing the video encoder to at least:

encode into the video sequence an indication of at least one image frame, which is the first image frame, in decoding order, of the independent sequence;

encode identifier values for the image frames according to a numbering scheme; and

reset the identifier value for the indicated first image frame of the independent sequence.

5. A video encoder according to claim 4, wherein the video encoder is configured to encode the indication into the video sequence as a separate flag included in the header of a slice.

6. A video encoder according to claim 4, wherein the video encoder is configured to encode into the video sequence an identifier value for the independent sequence.

7. A computer program product, stored on a non-transitory computer readable medium and executable in a data processing device, for encoding a video sequence comprising an independent sequence of image frames, wherein all motion-compensated temporal prediction references of the independent sequence refer only to image frames within said independent sequence, the computer program product comprising:

a computer program code for encoding into the video sequence an indication of at least one image frame, which is the first image frame, in decoding order, of the independent sequence;

a computer code for encoding identifier values for the image frames according to a numbering scheme; and

a computer code for resetting the identifier value for the indicated first image frame of the independent sequence.

8. A method for decoding a compressed video sequence, the method comprising:

decoding from the video sequence an indication of at least one image frame, which is the first image frame, in decoding order, of an independent sequence, wherein all motion-compensated temporal prediction references of

24

the independent sequence refer only to image frames within said independent sequence;

starting the decoding of the video sequence from said first image frame of the independent sequence, whereby the video sequence is decoded without prediction from any image frame decoded prior to said first image frame;

decoding identifier values for image frames according to a numbering scheme; and

resetting the identifier value for the indicated first image frame of the independent sequence.

9. A method according to claim 8, wherein the indication is a separate flag included in the header of a slice.

10. A video decoder comprising at least one processor and at least one memory including computer program code, the at least one memory and the computer program code configured to, with the at least one processor, cause the video decoder to decode a compressed video sequence, by causing the video decoder to at least:

decode from the video sequence an indication of at least one image frame, which is the first image frame, in decoding order, of an independent sequence, wherein all motion-compensated temporal prediction references of the independent sequence refer only to image frames within said independent sequence;

start the decoding of the video sequence from said first image frame of the independent sequence, whereby the video sequence is decoded without prediction from any image frame decoded prior to said first image frame; and

decode identifier values for image frames according to a numbering scheme; and

reset the identifier value for the indicated first image frame of the independent sequence.

11. A computer program product, stored on a non-transitory computer readable medium and executable in a data processing device, for decoding a compressed video sequence, the computer program product comprising:

a computer program code for decoding from the video sequence an indication of at least one image frame, which is the first image frame, in decoding order, of an independent sequence, wherein all motion-compensated temporal prediction references of the independent sequence refer only to image frames within said independent sequence;

a computer program code for starting the decoding of the video sequence from said first image frame of the independent sequence, whereby the video sequence is decoded without prediction from any image frame decoded prior to said first image frame;

a computer program code for decoding identifier values for image frames according to a numbering scheme; and

a computer program code for resetting the identifier value for the indicated first image frame of the independent sequence.

* * * * *

EXHIBIT 6



US007724818B2

(12) **United States Patent**
Hannuksela et al.

(10) **Patent No.:** **US 7,724,818 B2**
(45) **Date of Patent:** **May 25, 2010**

(54) **METHOD FOR CODING SEQUENCES OF PICTURES**

(75) Inventors: **Miska Hannuksela**, Tampere (FI);
Ye-Kui Wang, Tampere (FI)

(73) Assignee: **Nokia Corporation**, Espoo (FI)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1099 days.

(21) Appl. No.: **10/426,928**

(22) Filed: **Apr. 30, 2003**

(65) **Prior Publication Data**

US 2004/0218668 A1 Nov. 4, 2004

(51) **Int. Cl.**
H04N 7/12 (2006.01)

(52) **U.S. Cl.** **375/240.01**; 375/240.29

(58) **Field of Classification Search** 375/240.14,
375/240.01, 240.28, 240.25, 240.29, 240.1;
348/473; 709/201; 725/95

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,515,107 A * 5/1996 Chiang et al. 348/473
5,838,265 A 11/1998 Adolph
6,414,608 B1 7/2002 Nishida et al.
6,646,578 B1 11/2003 Au
7,072,393 B2 * 7/2006 Boice et al. 375/240.01
7,227,901 B2 * 6/2007 Joch et al. 375/240.29

2003/0012275 A1 * 1/2003 Boice et al. 375/240.01
2004/0008766 A1 1/2004 Wang et al.
2004/0008786 A1 1/2004 Boyce
2004/0010802 A1 * 1/2004 Visharam et al. 725/95
2004/0013202 A1 1/2004 Lainema
2004/0199565 A1 * 10/2004 Visharam et al. 709/201

FOREIGN PATENT DOCUMENTS

EP 0322123 6/1989
RU 2073913 2/1997
RU 2201654 3/2003
WO WO 02/15589 2/2002

OTHER PUBLICATIONS

"H.26L over IP and H.324 Framework"; ITU-Telecommunications Standardization Sector, Study Group 16 Question 6, Video Coding Experts Group (VCEG); Fourteenth Meeting, Santa Barbara, CA; Sep. 2001.

* cited by examiner

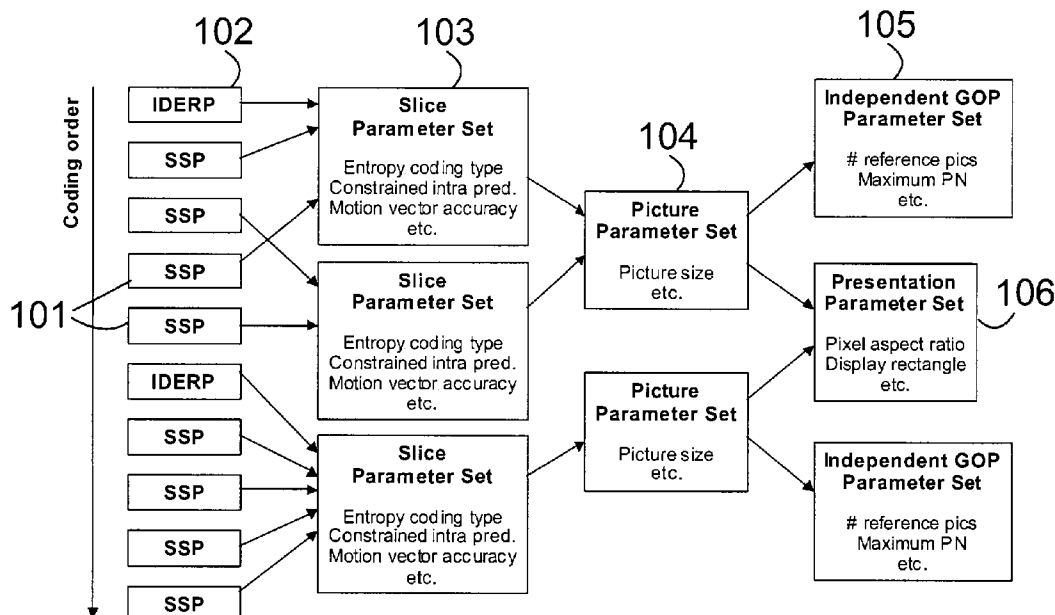
Primary Examiner—Gims S Philippe

(74) *Attorney, Agent, or Firm*—Ware, Fressola, Van Der Sluys & Adolphson LLP

(57) **ABSTRACT**

A method for encoding sequences of pictures into a bitstream, wherein parameters are defined in a parameter set and each picture comprises information of one or more slices. The method comprises defining parameters in a sequence parameter set; defining parameters in a picture parameter set; and defining at least one picture parameter in a slice header. The picture parameter remains unchanged at least in all slice headers of one picture.

23 Claims, 4 Drawing Sheets



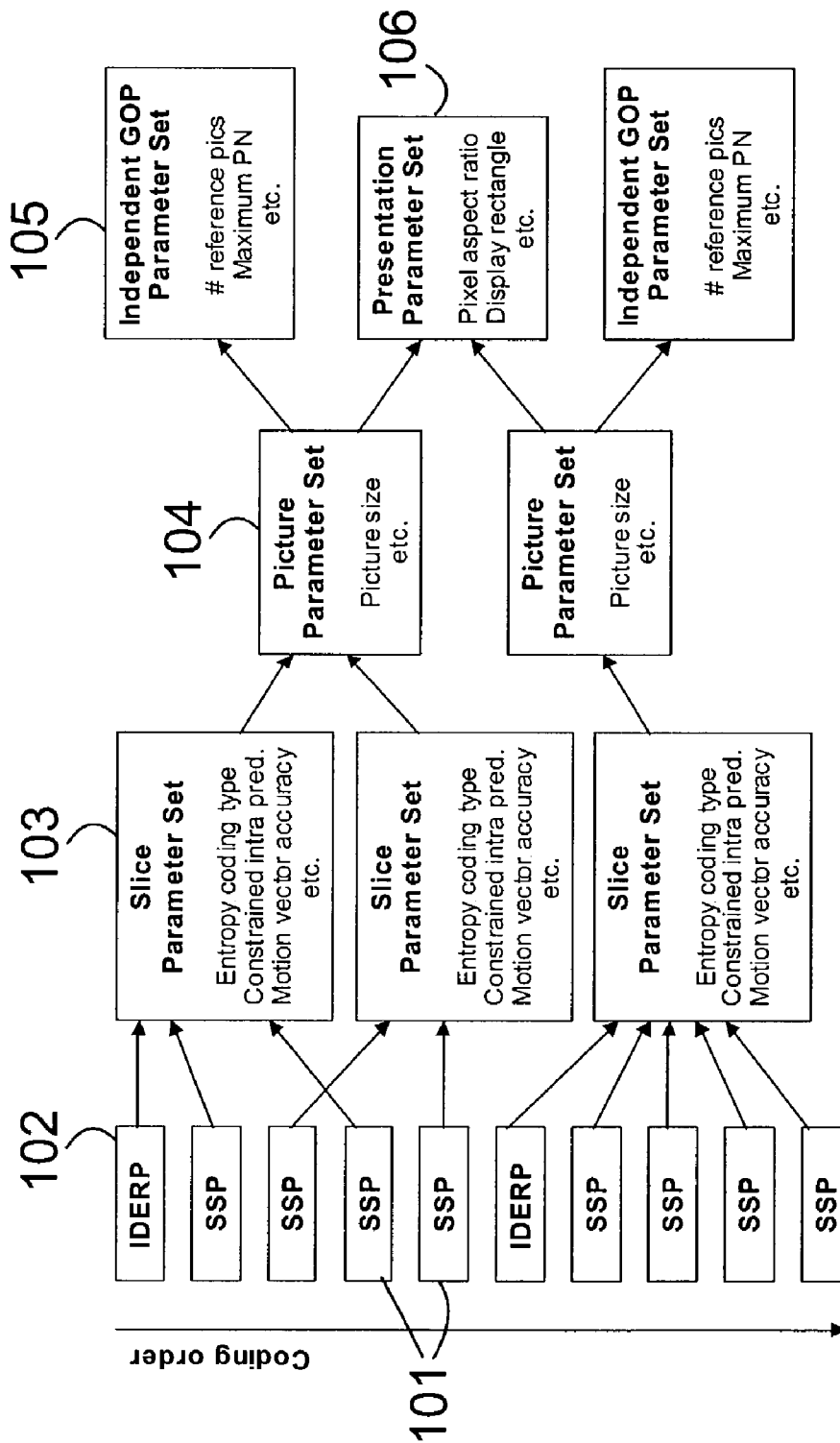


Fig. 1

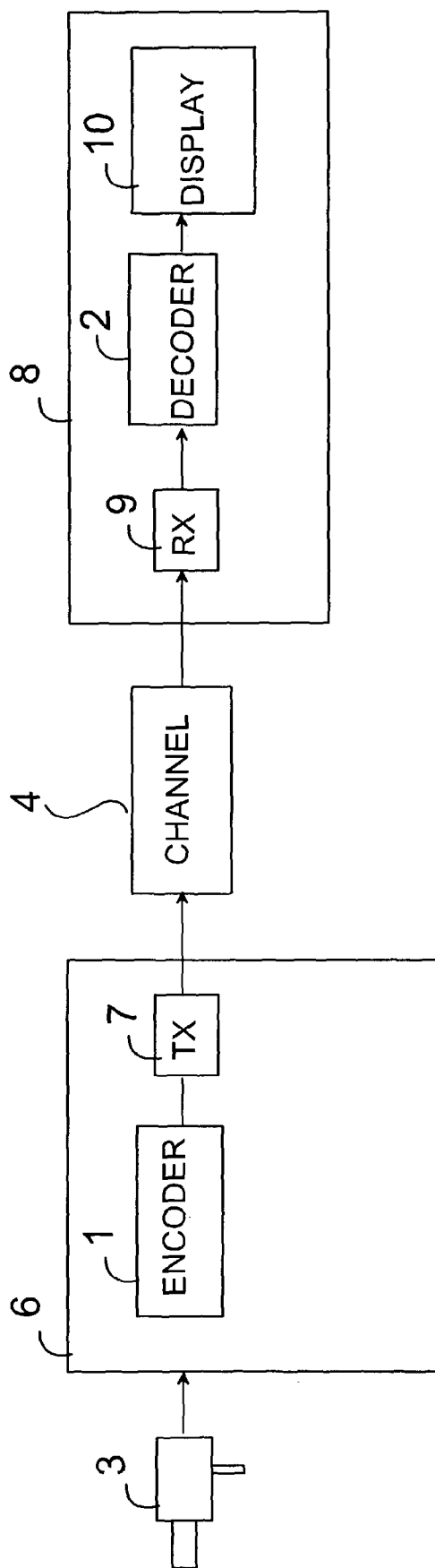


Fig. 2

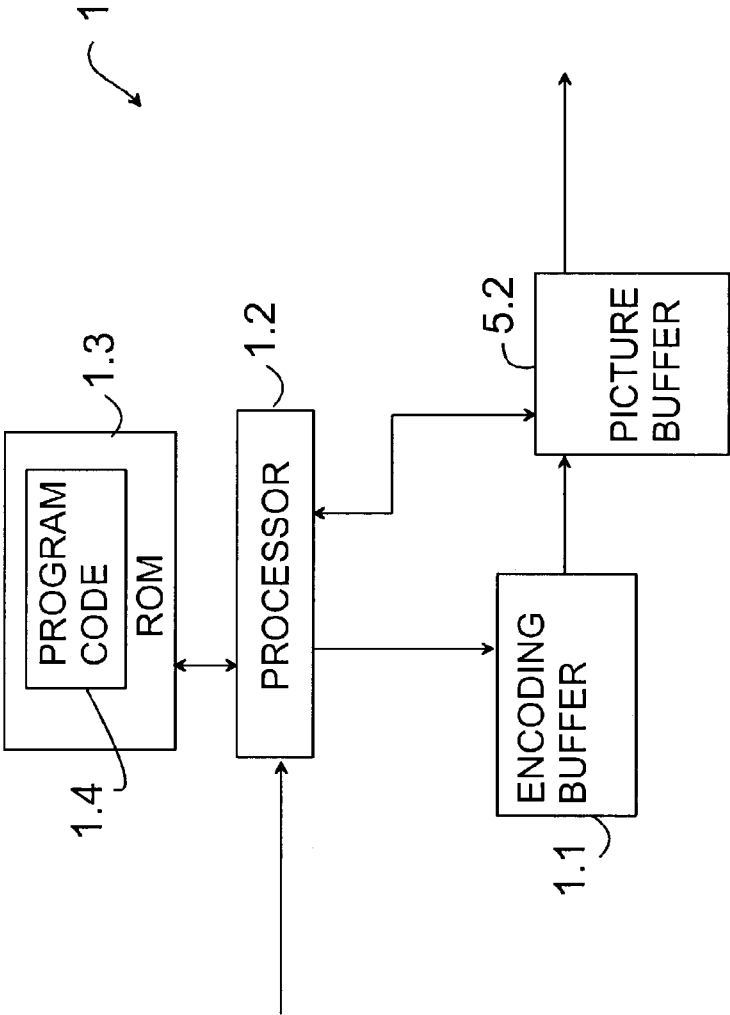


Fig. 3

2

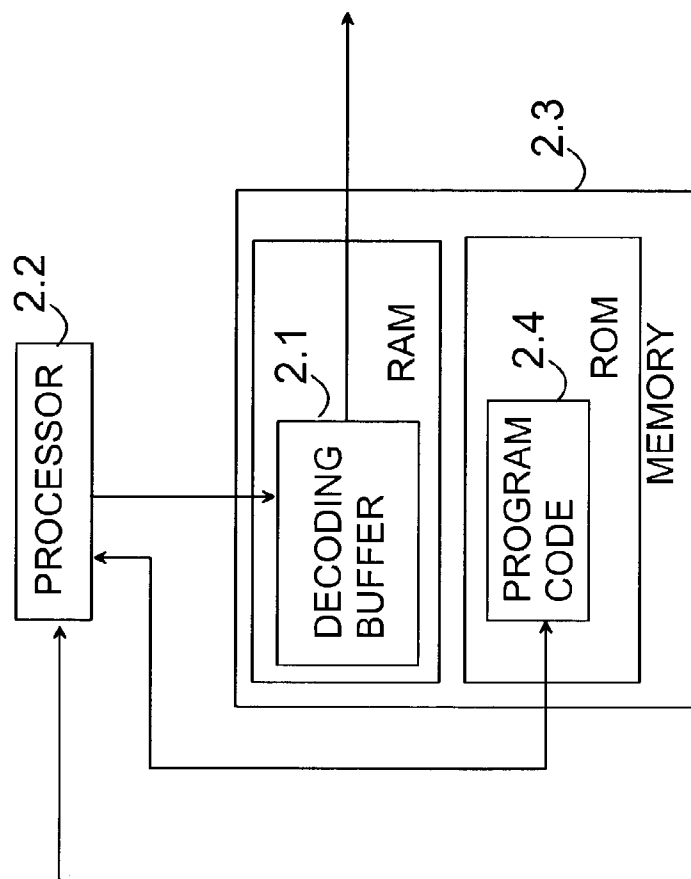


Fig. 4

METHOD FOR CODING SEQUENCES OF PICTURES

FIELD OF THE INVENTION

The present invention relates to a method for coding sequences of pictures into a bitstream, wherein parameters are defined in a parameter set and each picture comprises information of one or more slices. The invention further relates to a method for decoding sequences of pictures from a bitstream, wherein parameters are defined in a parameter set and each picture comprises information of one or more slices. The invention also relates to a system, transmitting device, receiving device, an encoder, a decoder, an electronic device, a software program, a storage medium, and a bitstream.

BACKGROUND OF THE INVENTION

Published video coding standards include ITU-T H.261, ITU-T H.263, ISO/IEC MPEG-1, ISO/IEC MPEG-2, and ISO/IEC MPEG-4 Part 2. These standards are herein referred to as conventional video coding standards.

There is a standardization effort going on in a Joint Video Team (JVT) of ITU-T and ISO/IEC. The work of JVT is based on an earlier standardization project in ITU-T called H.26L. The goal of the JVT standardization is to release the same standard text as ITU-T Recommendation H.264 and ISO/IEC International Standard 14496-10 (MPEG-4 Part 10). The draft standard is referred to as the JVT coding standard in this application, and the codec according to the draft standard is referred to as the JVT codec.

Video Communication Systems

Video communication systems can be divided into conversational and non-conversational systems. Conversational systems include video conferencing and video telephony. Examples of such systems include ITU-T Recommendations H.320, H.323, and H.324 that specify a video conferencing/telephony system operating in ISDN, IP, and PSTN networks respectively. Conversational systems are characterized by the intent to minimize the end-to-end delay (from audio-video capture to the far-end audio-video presentation) in order to improve the user experience.

Non-conversational systems include playback of stored content, such as Digital Versatile Disks (DVDs) or video files stored in a mass memory of a playback device, digital TV, and streaming.

In the following, some terms relating to video information are defined for clarity. A frame contains an array of luma samples and two corresponding arrays of chroma samples. A frame consists of two fields, a top field and a bottom field. A field is an assembly of alternate rows of a frame. A picture is either a frame or a field. A coded picture is either a coded field or a coded frame. In the JVT coding standard, a coded picture consists of one or more slices. A slice consists of an integer number of macroblocks, and a decoded macroblock corresponds to a 16×16 block of luma samples and two corresponding blocks of chroma samples. In the JVT coding standard, a slice is coded according to one of the following coding types: I (intra), P (predicted), B (bi-predictive), SI (switching intra), SP (switching predicted). A coded picture is allowed to contain slices of different types. All types of pictures can be used as reference pictures for P, B, and SP slices. The instantaneous decoder refresh (IDR) picture is a particular type of a coded picture including only slices with I or SI slice types. No subsequent picture can refer to pictures that are earlier than the IDR picture in decoding order. In some video coding standards, a coded video sequence is an entity containing all pictures in the bitstream before the end of a sequence mark. In the JVT coding standard, a coded video sequence is an entity

containing all coded pictures from an IDR picture (inclusive) to the next IDR picture (exclusive) in decoding order. In other words, a coded video sequence according to the JVT coding standard corresponds to a closed group of pictures (GOP) according to MPEG-2 video.

Conventional video coding standards have specified a structure for an elementary bitstream, i.e., a self-containing bitstream that decoders can parse. The bitstream has consisted of several layers, typically including several of the following: a sequence layer, a group of pictures (GOP) layer, a picture layer, a slice layer, a macroblock layer, and a block layer. The bitstream for each layer typically comprises a header and associated data.

The codec specification itself distinguishes conceptually between a video coding layer (VCL), and the network abstraction layer (NAL). The VCL contains the signal processing functionality of the codec, things such as transform, quantization, motion search/compensation, and the loop filter. It follows the general concept of most of today's video codecs, a macroblock-based coder that utilizes inter picture prediction with motion compensation, and transform coding of the residual signal. The output of the VCL are slices: a bit string that contains the macroblock data of an integer number of macroblocks, and the information of the slice header (containing the spatial address of the first macroblock in the slice, the initial quantization parameter, and similar). Macroblocks in slices are ordered in scan order unless a different macroblock allocation is specified, using the so-called Flexible Macroblock Ordering syntax. In-picture prediction is used only within a slice.

The NAL encapsulates the slice output of the VCL into Network Abstraction Layer Units (NALUs), which are suitable for the transmission over packet networks or the use in packet oriented multiplex environments. All NAL units relating to a certain picture form an access unit. JVT's Annex B defines an encapsulation process to transmit such NALUs over byte-stream oriented networks. A stream of NAL units does not form an elementary bitstream as such because there are no start codes in NAL units, but rather NAL units have to be framed with start codes according to Annex B of the JVT coding standard to form an elementary bitstream.

The optional reference picture selection mode of H.263 and the NEWPRED coding tool of MPEG-4 Part 2 enable selection of the reference frame for motion compensation per each picture segment, e.g., per each slice in H.263. Furthermore, the optional Enhanced Reference Picture Selection mode of H.263 and the JVT coding standard enable selection of the reference frame for each macroblock separately.

Parameter Set Concept

The JVT coding standard contains headers at slice layer and below, but it does not include picture, GOP, or sequence headers. Instead, a concept of a parameter set, introduced in ITU-T document VCEG-N55, replaces such headers. An instance of a parameter set includes all picture, GOP, and sequence level data such as picture size, display window, optional coding modes employed, macroblock allocation map, and others. Each parameter set instance includes a unique identifier. Each slice header includes a reference to a parameter set identifier, and the parameter values of the referred parameter set are used when decoding the slice. Parameter sets decouple the transmission and decoding order of infrequently changing picture, GOP, and sequence level data from sequence, GOP, and picture boundaries. Parameter sets can be transmitted out-of-band using a reliable transmission protocol as long as they are decoded before they are referred. If parameter sets are transmitted in-band, they can be repeated multiple times to improve error resilience compared to conventional video coding schemes. Preferably the parameter sets are transmitted at a session set-up time. However, in

some systems, mainly broadcast ones, reliable out-of-band transmission of parameter sets is not feasible, but rather parameter sets are conveyed in-band in Parameter Set NAL units.

In order to be able to change picture parameters (such as the picture size), without having the need to transmit Parameter Set updates synchronously to the slice packet stream, the encoder and decoder can maintain a list of more than one Parameter Set. Each slice header contains a codeword that indicates the Parameter Set to be used.

This mechanism allows decoupling of the transmission of the Parameter Sets from the packet stream, and transmit them by external means, e.g. as a side effect of the capability exchange, or through a (reliable or unreliable) control protocol. It may even be possible that they are never transmitted but are fixed by an application design specification.

There are some disadvantages with pre-defined parameter sets. First, if there is a need to transmit many parameter set instances in the beginning of a session, the out-of-band method may become overburdened or the beginning latency of the session will be too long. Second, in systems lacking feasible mechanisms for reliable out-of-band transmission of parameter sets, in-band transport of Parameter Set NAL units is not reliable. Third, for broadcast applications, since the parameter sets information should be transmitted frequently to allow new users join during the broadcast process, redundant transmission of all the active parameter set instances is costly from bit-rate point of view.

Transmission of Multimedia Streams

A multimedia streaming system consists of a streaming server and a number of players, which access the server via a network. The network is typically packet-oriented and provides little or no means to guarantee quality of service. The players fetch either pre-stored or live multimedia content from the server and play it back in real-time while the content is being downloaded.

The type of communication can be either point-to-point or multicast. In point-to-point streaming, the server provides a separate connection for each player. In multicast streaming, the server transmits a single data stream to a number of players, and network elements duplicate the stream only if it is necessary.

When a player has established a connection to a server and requested for a multimedia stream, the server begins to transmit the desired stream. The player does not start playing the stream back immediately, but rather it typically buffers the incoming data for a few seconds. Herein, this buffering is referred to as initial buffering. Initial buffering helps to maintain pauseless playback, because, in case of occasional increased transmission delays or network throughput drops, the player can decode and play buffered data.

SUMMARY OF THE INVENTION

One aim of the present invention is to avoid the problems of prior art and provide a more efficient method to transmit parameters relating to picture information. According to an advantageous embodiment of the present invention, the following method for parameter set signalling is provided:

A relatively large number of frequently used parameter set instances and their IDs are pre-defined and stored both in an encoding device and in a decoding device. When the communication starts, these pre-defined parameter sets need not be transmitted. Only the parameter sets not included in the pre-defined ones have to be transmitted, preferably at the session beginning, or transmitted in parameter set NAL units later if

necessary. The system could include a definition of default parameter sets for each profile and level.

According to the invention, there exist at least two kinds of parameter sets: one for the sequence (sequence parameter set) and one for the pictures (picture parameter set).

For applications without feedback channels (digital TV) or with feedback channels of very limited use (e.g. multicast streaming with a huge number of receivers), the set of pre-defined parameter sets should be as complete as possible, from the point of view of possible systems and application scenarios. Therefore, it may be possible that no parameter sets need to be transmitted. Once a parameter set NAL unit is needed, it should be transmitted multiple times to ensure that it is correctly received.

Another inventive concept of the present invention splits the parameter set structure to multiple parameter set structures according to the persistency and target of parameters. In particular, the invention includes the concepts of sequence parameter set and picture parameter set. The selection of a correct parameter set for each parameter depends on the probable and allowed changing rate of the parameter. Parameters whose value may change in every slice or whose value is likely to change in every picture, are included in the slice header. If it is probable that a parameter remains unchanged in multiple pictures but is allowed to change in every picture, such parameter is included in a picture parameter set. Such parameters which are not allowed to change in a coded video sequence are included in the sequence parameter set. Some non-restrictive examples of such parameters are picture order count, frame number and identifier of an independently decodable picture.

In the following description the invention is described by using encoder-decoder based system, but it is obvious that the invention can also be implemented in systems in which the video signals are stored. The stored video signals can be either uncoded signals stored before encoding, encoded signals stored after encoding, or decoded signals stored after encoding and decoding process. For example, an encoder produces bitstreams. A file system receives audio and/or video bitstreams which are encapsulated e.g. in decoding order and stored as a file.

The encoding method according to the present invention is primarily characterized in that the method comprises:

- defining parameters in a sequence parameter set;
- defining parameters in a picture parameter set; and
- defining at least one picture parameter in a slice header, the picture parameter remaining unchanged at least in all slice headers of one picture.

The decoding method according to the present invention is primarily characterized in that the method comprises:

- recognizing a sequence parameter set and forming at least one sequence parameter pertaining to a sequence using the parameter set;
- recognizing a picture parameter set and forming at least one first picture parameter pertaining to a picture using the parameter set; and
- forming at least one second picture parameter using information of a slice header, the at least one second picture parameter remaining unchanged at least in all slice headers of one picture

using the at least one second picture parameter in decoding.

The encoder according to the present invention is primarily characterized in that it comprises:

- means for defining parameters in a sequence parameter set;
- means for defining parameters in a picture parameter set; and

5

means for defining at least one picture parameter in a slice header, the picture parameter remaining unchanged at least in all slice headers of one picture.

The decoder according to the present invention is primarily characterized in that the it comprises:

means for recognizing a sequence parameter set and forming at least one sequence parameter pertaining to a sequence using the parameter set;

means for recognizing a picture parameter set and forming at least one first picture parameter pertaining to a picture using the parameter set;

means for forming at least one second picture parameter using information of a slice header, the at least one second picture parameter remaining unchanged at least in all slice headers of one picture; and

means for using the at least one second picture parameter in decoding.

The system according to the present invention is primarily characterized in that the encoder comprises:

means for defining parameters in a sequence parameter set;

means for defining parameters in a picture parameter set; and

means for defining at least one picture parameter in a slice header, the picture parameter remaining unchanged at least in all slice headers of one picture;

and the decoder comprises:

means for recognizing a sequence parameter set and forming at least one sequence parameter pertaining to a sequence using the parameter set;

means for recognizing a picture parameter set and forming at least one first picture parameter pertaining to a picture using the parameter set;

means for forming at least one second picture parameter using information of a slice header, the at least one second picture parameter remaining unchanged at least in all slice headers of one picture; and

means for using the at least one second picture parameter in decoding.

The transmitting device according to the present invention is primarily characterized in that the encoder comprises:

means for defining parameters in a sequence parameter set; means for defining parameters in a picture parameter set; and

means for defining at least one picture parameter in a slice header, the picture parameter remaining unchanged at least in all slice headers of one picture.

The receiving device according to the present invention is primarily characterized in that the decoder comprises:

means for recognizing a sequence parameter set and forming at least one sequence parameter pertaining to a sequence using the parameter set;

means for recognizing a picture parameter set and forming at least one first picture parameter pertaining to a picture using the parameter set;

means for forming at least one second picture parameter using information of a slice header, the at least one second picture parameter remaining unchanged at least in all slice headers of one picture; and

means for using the at least one second picture parameter in decoding.

The bitstream according to the present invention is primarily characterized in that it comprises:

encoded pictures;

sequence parameters in a sequence parameter set;

picture parameters in a picture parameter set;

information of a slice comprising a slice header; and

6

at least one picture parameter in the slice header, the picture parameter remaining unchanged at least in all slice headers of one encoded picture.

The software program for encoding according to the present invention is primarily characterized in that the software program comprises:

defining parameters in a sequence parameter set;

defining parameters in a picture parameter set; and

defining at least one picture parameter in a slice header, the picture parameter remaining unchanged at least in all slice headers of one picture.

The software program for decoding according to the present invention is primarily characterized in that the software program comprises: the software program comprising:

recognizing a sequence parameter set and forming at least one sequence parameter pertaining to a sequence using the parameter set;

recognizing a picture parameter set and forming at least one first picture parameter pertaining to a picture using the parameter set;

forming at least one second picture parameter using information of a slice header, the at least one second picture parameter remaining unchanged at least in all slice headers of one picture; and

using the at least one second picture parameter in decoding.

The storage medium including the software program for encoding according to the present invention is primarily characterized in that the software program comprises machine executable steps for:

defining parameters in a sequence parameter set;

defining parameters in a picture parameter set; and

defining at least one picture parameter in a slice header, the picture parameter remaining unchanged at least in all slice headers of one picture.

The storage medium including the software program for decoding according to the present invention is primarily characterized in that the software program comprises machine executable steps for:

recognizing a sequence parameter set and forming at least one sequence parameter pertaining to a sequence using the parameter set;

recognizing a picture parameter set and forming at least one first picture parameter pertaining to a picture using the parameter set;

forming at least one second picture parameter using information of a slice header, the at least one second picture parameter remaining unchanged at least in all slice headers of one picture; and

using the at least one second picture parameter in decoding.

The present invention improves compression efficiency. It is likely that the number of picture parameter sets is larger than the number of sequence parameter sets and the frequency of updating picture parameter sets is higher than the frequency of updating sequence parameter sets. Thus, if there were a single parameter set structure, many sequence-level parameters that remained unchanged in the previous picture parameter set (in decoding order) should be repeated. Including picture and sequence level parameters in different syntax structures helps to avoid this problem.

The present invention clarifies the persistency rules of parameter values. Certain parameter values, such as the picture size, shall remain unchanged within the sequence. Other parameter values may change from picture to picture. If there were a single parameter set structure, there should be semantic restrictions, which parameter values must not change within a sequence even though the referred parameter set may change within the sequence. Now that the sequence param-

eter set structure is specified, it is clear that all picture parameter sets that are referred to within a sequence must refer to the same sequence parameter set. Moreover, it is clear that all slices of a picture must refer to the same picture parameter set.

DESCRIPTION OF THE DRAWINGS

FIG. 1 presents an illustration of dependencies between NAL packets and parameter set instances,

FIG. 2 depicts an advantageous embodiment of the system according to the present invention,

FIG. 3 depicts an advantageous embodiment of the encoder according to the present invention, and

FIG. 4 depicts an advantageous embodiment of the decoder according to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

In an advantageous embodiment of the present invention four parameter set structures are defined: an independent GOP parameter set or a sequence parameter set, a picture parameter set, a slice parameter set, and, optionally, a presentation parameter set. Some reasons for the definition of the different parameter sets are as follows:

First, certain parameter values must remain unchanged throughout an independent GOP or within a picture. For example, the number of picture “slots” in a multi-picture buffer must not be changed during an independent GOP. Otherwise, it would be unclear how the multi-picture buffering process operates. A decoder implementation has to ensure that all the slices of a particular picture refer to the same picture parameter set. Similarly, all the slices of an independent GOP must refer to the same independent GOP parameter set. Otherwise, decoders should infer data loss or corruption. Obtaining the same functionality with one joint parameter set requires that the decoder checks that individual parameter values remain the same within a picture or within an independent GOP.

Second, a compact syntax for parameter sets is advantageous to save bits in Parameter set NAL units. Thus, it makes sense to separate independent GOP and picture parameter sets from more frequently updated slice parameter sets.

Third, display-related parameter values do not affect decoding of a coded video stream.

FIG. 1 presents an illustration of dependencies between NAL packets and parameter set instances. An arrow indicates a reference based on a parameter set identifier. The starting point of an arrow is the entity, which refers to the parameter set instance where the arrow points. The ending point of an arrow is the owner of the parameter set identifier. A slice (or a Single slice, SSP) 101, slice data partition, and IDR NAL units (IDR RP) (or independent decoder refresh NAL packets) 102 always refer to a slice parameter set 103. A slice parameter set 103 refers to a picture parameter set 104, and a picture parameter set 104 refers to both an independent GOP parameter set 105 and a presentation parameter set 106, or to a sequence parameter set. All the slices of a picture should refer to the same picture parameter set and all the slices of a sequence should refer to the same sequence parameter set.

Next, the different parameter sets according to an advantageous embodiment of the present invention will be described in more detail.

Independent GOP Parameter Set

The start of an independent GOP is identified by an Instantaneous Decoder Refresh NAL packet (IDRNP). An advantageous syntax of the packet is presented in Table 1.

TABLE 1

Independent GOP Parameter Set:	
parameter_set_id	
Profile	
Level	
Version	
log2_max_picture_number_minus_4	
number_of_reference_picture_buffers_minus_1	
required_picture_number_update_behavior	

The meaning of the fields of the packet will now be explained. The profile field defines the coding profile in use; the level field defines the level in use within the profile; the version field defines the version in use within the profile and the level. The next field, log2_max_picture_number_minus_4, specifies the MAX_PN constant used in picture number related arithmetic. MAX_PN is calculated by raising 2 to the power of the value of this field (log2_max_picture_number_minus_4) and decrementing the calculated power of 2 by 1. The field number_of_reference_picture_buffers_minus_1 defines the total number of short- and long-term picture buffers in the multi-picture buffer. If the value of the required_picture_number_update_behavior field is 1, a specific decoder behavior in case of missing picture numbers is mandated. However, this is outside the scope of the present invention wherein it is not described in this application.

Profile, level, and version are very likely to remain unchanged in an independent GOP. Therefore, they are included in the independent GOP parameter set.

A change in reference picture buffering controls, i.e., MAX_PN, number of reference picture buffers, and required picture number update behavior, would cause an undefined decoder state. A change of these parameters in the middle of an independent GOP would not bring any benefits. Hence, the number of reference picture buffers is informed in the independent GOP parameter set. A range from 0 to 15 is considered to be a practical minimum for a picture number, and therefore smaller values are should be avoided.

Picture Parameter Set

The picture parameter set includes the fields presented in Table 2.

TABLE 2

Picture Parameter Set	
parameter_set_id of a picture_parameter_set	
parameter_set_id of an independent GOP parameter set	
parameter_set_id of a presentation parameter set	
picture_width_in_MBs_minus_1	
picture_height_in_MBs_minus_1	

The first parameter_set_id field indicates the picture_parameter_set in question. The other two parameter sets indicate the independent GOP parameter set and the presentation parameter set which will be used together with the picture parameter set in encoding and decoding of the slices referring to the picture parameter set. More than one picture parameter sets can refer to the same independent GOP parameter set and/or to the same presentation parameter set. The picture_width_in_MBs_minus_1 field and the picture_height_in_MBs_minus_1 field define the size of the picture.

In some embodiments the picture_width_in_MBs_minus_1 field and picture_height_in_MBs_minus_1 field could reside in the independent GOP parameter set as well.

The presentation parameter set is referred to in picture level, as it is necessary to allow changes in the presentation parameters on picture-by-picture basis. For example, the presentation parameter set signals the display rectangle of the reconstructed pictures, which is directly related to the coded picture size signaled in the picture parameter set.

Slice Parameter Set

The slice parameter set includes the fields presented in Table 3.

TABLE 3

Slice Parameter Set	
parameter_set_id of a slice parameter set	
parameter_set_id of a picture parameter set	
entropy_coding	
motion_resolution	
constrained_intra_prediction_flag	
multiple_prediction_frames_flag	

The entropy_coding field indicates the VLC coding type of WD2. If the value of the entropy_coding field equals to zero it indicates that non-arithmetic VLC coding of WD2 is used, whereas value one indicates that arithmetic coding VLC coding of WD2 is used.

The motion_resolution field gives information on motion resolution. If the value of the motion_resolution field equals to zero a 1/4-sample motion resolution is in use, and if the value equals to one a 1/8-sample motion resolution is in use.

The constrained_intra_prediction_flag is used to provide information on prediction mode. If the constrained_intra prediction flag equals to zero normal intra prediction is used, whereas one stands for the constrained intra prediction. In the constrained intra prediction mode, no intra prediction is done from inter macroblocks.

The multiple_prediction_frames_flag gives information on reference picture usage for motion compensation. The multiple_prediction_frames_flag equal to zero signals that only the previous coded picture in coding order is used as a reference picture for motion compensation in P- and SP-frames and the most recent previous decoded and subsequent decoded pictures are used for motion compensation in B-frames. Otherwise, the reference picture(s) for prediction must be signaled for each macroblock.

The selected parameters for the slice parameter set were such that no reason to restrict their values to be unchanged in an entire independent GOP was found. Instead, it might be advantageous to allow changes in parameter values even within a picture. Examples of possible advantages include:

Use of multiple reference pictures. It may make sense to restrict the number of reference pictures to one for certain parts of the picture. For example, in wireline video conferencing equipment, the encoder may treat the center of the picture better than the edges.

Entropy coding type. It may be possible to mix multiple coded video streams to one without decoding the streams. This can be beneficial in a multi-point control unit (MCU) of a video conferencing system, for example. One endpoint may use a different entropy coding method than another one. Thus, the "mixed" stream would contain multiple entropy coding types in a same "mixed" coded picture.

Presentation Parameter Set

The presentation parameter set includes the fields presented in Table 4.

TABLE 4

Presentation Parameter Set	
parameter_set_id of a presentation parameter set	
pixel_aspect_ratio_width	
pixel_aspect_ratio_height	
display_rectangle_offset_top	
display_rectangle_offset_left	
display_rectangle_offset_bottom	
display_rectangle_offset_right	

The displayed pixel aspect ratio should be pixel_aspect_ratio_width:pixel_aspect_ratio_height. The parameter values shall be relatively prime. Value 0 is preferably forbidden. The display_rectangle_offset_top field, display_rectangle_offset_left field, display_rectangle_offset_bottom field, and display_rectangle_offset_right field define the rectangle to be displayed from the coded picture. Sample units are used.

Parameter set NAL units can be used to update values of parameter sets for a video stream. The parameter update packets can be transmitted as Network Abstraction Layer Packets (NALP). A NALP consists of a NALP header (NALPH) and a NALP payload (NALPP). The NALPH is the first byte of the NALP. The NALPH itself distinguishes different NALP types and includes one bit (EI flag) indicating the presence of errors in the NALPP following the NALPH. EI flag set to 0 means that there is no known error in the following payload whereas a 1 indicates a corrupted payload and/or a corrupted NALP type.

A Parameter set NAL unit becomes valid synchronously with the decoding process according to the specific type of the parameter set NAL unit as follows: An update of an independent GOP parameter set takes place just before the first slice of the next independent GOP (i.e., the next IDERP NAL packet) is decoded. An update of picture and presentation parameter sets takes place just before the first slice of the next picture is decoded. An update of a slice parameter set takes place substantially immediately if at least one of the contained parameters is changed. However, slice parameter set identifier used in a picture may not be redefined with different parameter values within the same coded picture.

For certain broadcast applications, such as digital TV and multicast streaming with a huge number of possible receivers, new independent GOP parameter sets, which fall out of the scope of the pre-defined ones, are likely to be updated before each independent GOP is decoded, regardless of whether or not some of the contained parameters are changed, to enable decoding in terminals that just started receiving the coded data. Other parameter sets are updated before they are referred to in the coded data.

An advantageous embodiment of a system in which the present invention can be implemented is presented in FIG. 2. The pictures to be encoded can be, for example, pictures of a video stream from a video source 3, e.g. a camera, a video recorder, etc. The pictures (frames) of the video stream can be divided into smaller portions such as slices. The slices can further be divided into macroblocks and blocks. The encoder 1 selects proper parameter sets for use with the encoding process and sends the selected parameter sets to the decoder 2. The decoder stores the parameter sets to memory 2.3 (FIG. 4). If the decoder already has a correct parameter set, it is not necessary to transmit such parameter set to the decoder 2 but only an indication of the correct parameter set.

In the encoder 1 the video stream is encoded to reduce the information to be transmitted via a transmission channel 4, or to a storage media (not shown). Pictures of the video stream

11

are input to the encoder 1. The encoder has an encoding buffer 1.1 (FIG. 3) for temporarily storing some of the pictures to be encoded. The encoder 1 also includes a memory 1.3 and a processor 1.2 in which the encoding tasks according to the invention can be applied. The memory 1.3 and the processor 1.2 can be common with the transmitting device 6 or the transmitting device 6 can have another processor and/or memory (not shown) for other functions of the transmitting device 6. The encoder 1 performs motion estimation and/or some other tasks to compress the video stream. In motion estimation similarities between the picture to be encoded (the current picture) and a previous and/or latter picture are searched. If similarities are found the compared picture or part of it can be used as a reference picture for the picture to be encoded. In the JVT coding standard the display order and the decoding order of the pictures are not necessarily the same, wherein the reference picture has to be stored in a buffer (e.g. in the encoding buffer 1.1) as long as it is used as a reference picture. The encoder 1 may also insert information on display order of the pictures into the bitstream.

From the encoding process the encoded pictures are moved to an encoded picture buffer 5.2, if necessary. The encoded pictures are transmitted from the encoder 1 to the decoder 2 via the transmission channel 4. In the decoder 2 the encoded pictures are decoded to form uncompressed pictures corresponding as much as possible to the encoded pictures. Each decoded picture is buffered in the DPB 2.1 of the decoder 2 unless it is displayed substantially immediately after the decoding and is not used as a reference picture. Preferably both the reference picture buffering and the display picture buffering are combined and they use the same decoded picture buffer 2.1. This eliminates the need for storing the same pictures in two different places thus reducing the memory requirements of the decoder 2.

The decoder 2 also includes a memory 2.3 and a processor 2.2 in which the decoding tasks according to the invention can be applied. The memory 2.3 and the processor 2.2 can be common with the receiving device 8 or the receiving device 8 can have another processor and/or memory (not shown) for other functions of the receiving device 8.

It is obvious that the present invention is not limited solely to the above described embodiments but it can be modified within the scope of the appended claims.

The invention claimed is:

1. A method for encoding sequences of pictures into a bitstream, wherein parameters are defined in a parameter set and each picture comprises information of one or more slices, the method comprising:

defining, in an encoder, parameter values in a sequence parameter set for a sequence of pictures;
defining, in the encoder, parameter values in a picture parameter set for a picture; and
defining, in the encoder, at least one picture parameter value in a slice header, the picture parameter value remaining unchanged at least in all slice headers of one picture.

2. The method according to claim 1, wherein the slice header is included with a reference to a picture parameter set.

3. The method according to claim 1, wherein the picture parameter set is included with a reference to a sequence parameter set.

4. The method according to claim 1, wherein the picture parameter value is selected from a group of information of a picture order count and a frame number.

5. The method according to claim 1, wherein the sequence parameter set and picture parameter set are transmitted less often than once per each picture.

12

6. A method for decoding sequences of pictures from a bitstream, wherein parameters are defined in a parameter set and each picture comprises information of one or more slices, the method comprising:

recognizing, in a decoder, a sequence parameter set and forming at least one sequence parameter pertaining to a sequence using the parameter set;

recognizing, in the decoder, a picture parameter set and forming at least one first picture parameter value pertaining to a picture using the parameter set;

forming, in the decoder, at least one second picture parameter value using information of a slice header, the at least one second picture parameter value remaining unchanged at least in all slice headers of one picture; and
using, in the decoder, the at least one second picture parameter value in decoding.

7. The method according to claim 6, wherein the slice header is included with a reference to a picture parameter set.

8. The method according to claim 6, wherein the picture parameter set is included with a reference to a sequence parameter set.

9. The method according to claim 6, wherein the second picture parameter value is selected from a group of information of a picture order count and a frame number.

10. An encoder for encoding sequences of pictures into a bitstream, wherein parameters are defined in a parameter set and each picture comprises information of one or more slices, the encoder comprising:

a processor configured for:

defining parameter values in a sequence parameter set for a sequence of pictures;

defining parameter values in a picture parameter set for a picture; and

defining at least one picture parameter value in a slice header, the picture parameter value remaining unchanged at least in all slice headers of one picture.

11. A decoder for decoding sequences of pictures from a bitstream, wherein parameters are defined in a parameter set and each picture comprises information of one or more slices, the decoder comprising:

a processor configured for:

recognizing a sequence parameter set for a sequence of pictures and forming at least one sequence parameter pertaining to a sequence using the parameter set for a picture;

recognizing a picture parameter set and forming at least one first picture parameter value pertaining to a picture using the parameter set;

forming at least one second picture parameter value using information of a slice header, the at least one second picture parameter value remaining unchanged at least in all slice headers of one picture; and

using the at least one second picture parameter value in decoding.

12. A system comprising an encoder for encoding sequences of pictures into a bitstream, a decoder for decoding sequences of pictures from the bitstream, wherein parameters are defined in a parameter set and each picture comprises information of one or more slices, the encoder comprising:

a processor configured for:

defining parameter values in a sequence parameter set for a sequence of pictures;

defining parameter values in a picture parameter set for a picture; and

defining at least one picture parameter value in a slice header, the picture parameter value remaining unchanged at least in all slice headers of one picture;

13

and the decoder comprising:

a processor configured for:

recognizing a sequence parameter set for a sequence of pictures and forming at least one sequence parameter pertaining to a sequence using the parameter set;

recognizing a picture parameter set for a picture and forming at least one first picture parameter value pertaining to a picture using the parameter set;

forming at least one second picture parameter value using information of a slice header, the at least one second picture parameter value remaining unchanged at least in all slice headers of one picture; and

using the at least one second picture parameter value in decoding.

13. The system according to claim 12, wherein a parameter set is stored in both the encoder and the decoder.

14. A transmitting device comprising an encoder for encoding sequences of pictures into a bitstream, wherein parameters are defined in a parameter set and each picture comprises information of one or more slices, the encoder comprising:

a processor configured for:

defining parameter values in a sequence parameter set for a sequence of pictures;

defining parameter values in a picture parameter set for a picture; and

defining at least one picture parameter value in a slice header, the picture parameter value remaining unchanged at least in all slice headers of one picture.

15. A receiving device comprising a decoder for decoding sequences of pictures from a bitstream, wherein parameters are defined in a parameter set and each picture comprises information of one or more slices, the decoder comprising:

a processor configured for:

recognizing a sequence parameter set for a sequence of pictures and forming at least one sequence parameter pertaining to a sequence using the parameter set;

recognizing a picture parameter set for a picture and forming at least one first picture parameter value pertaining to a picture using the parameter set;

forming at least one second picture parameter value using information of a slice header, the at least one second picture parameter value remaining unchanged at least in all slice headers of one picture; and

using the at least one second picture parameter value in decoding.

16. A bitstream comprising:

encoded pictures;

sequence parameters in a sequence parameter set for a sequence of pictures;

picture parameter values in a picture parameter set for a picture;

information of a slice comprising a slice header; and

at least one picture parameter value in the slice header, the picture parameter value remaining unchanged at least in all slice headers of one encoded picture.

17. The bitstream according to claim 16, wherein the slice header is included with a reference to a picture parameter set.

18. The bitstream according to claim 16, wherein the picture parameter set is included with a reference to a sequence parameter set.

19. The bitstream according to claim 16, wherein the picture parameter value is selected from a group of information of a picture order count and a frame number.

14

20. A computer readable storage medium embedded with a computer program comprising programming code for execution on a processor, the programming code for encoding sequences of pictures into a bitstream, wherein parameters are defined in a parameter set and each picture comprises information of one or more slices, and the programming code for:

defining parameter values in a sequence parameter set for a sequence of pictures;

defining parameter values in a picture parameter set for a picture; and

defining at least one picture parameter value in a slice header, the picture parameter value remaining unchanged at least in all slice headers of one picture.

21. A computer readable storage medium embedded with a computer program comprising programming code for execution on a processor, the programming code for decoding sequences of pictures from a bitstream, wherein parameters are defined in a parameter set and each picture comprises information of one or more slices, and the programming code for:

recognizing a sequence parameter set for a sequence of pictures and forming at least one sequence parameter pertaining to a sequence using the parameter set;

recognizing a picture parameter set for a picture and forming at least one first picture parameter value pertaining to a picture using the parameter set;

forming at least one second picture parameter value using information of a slice header, the at least one second picture parameter value remaining unchanged at least in all slice headers of one picture; and

using the at least one second picture parameter value in decoding.

22. An encoder for encoding sequences of pictures into a bitstream, wherein parameters are defined in a parameter set and each picture comprises information of one or more slices, the encoder comprising:

means for defining parameter values in a sequence parameter set for a sequence of pictures;

means for defining parameter values in a picture parameter set for a picture; and

means for defining at least one picture parameter value in a slice header, the picture parameter value remaining unchanged at least in all slice headers of one picture.

23. A decoder for decoding sequences of pictures from a bitstream, wherein parameters are defined in a parameter set and each picture comprises information of one or more slices, the decoder comprising:

means for recognizing a sequence parameter set for a sequence of pictures and forming at least one sequence parameter pertaining to a sequence using the parameter set for a picture;

means for recognizing a picture parameter set and forming at least one first picture parameter value pertaining to a picture using the parameter set;

means for forming at least one second picture parameter value using information of a slice header, the at least one second picture parameter value remaining unchanged at least in all slice headers of one picture; and

means for using the at least one second picture parameter value in decoding.

EXHIBIT 7



US006950469B2

(12) **United States Patent**
Karczewicz et al.

(10) **Patent No.:** **US 6,950,469 B2**
(45) **Date of Patent:** **Sep. 27, 2005**

(54) **METHOD FOR SUB-PIXEL VALUE INTERPOLATION**

GB 2205707 12/1988

OTHER PUBLICATIONS

(75) Inventors: **Marta Karczewicz**, Irving, TX (US);
Antti Olli Hallapuro, Tampere (FI)

Lappalainen, Ville, Performance analysis of Intel MMX technology for an H.263 video H.263 video encoder, 1998, ACM Press, pp. 309–314.*

(73) Assignee: **Nokia Corporation**, Espoo (FI)

* cited by examiner

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 688 days.

Primary Examiner—Chris Kelley
Assistant Examiner—Matthew Haney
(74) *Attorney, Agent, or Firm*—Perman & Green, LLP

(21) Appl. No.: **09/954,608**

(57) **ABSTRACT**

(22) Filed: **Sep. 17, 2001**

(65) **Prior Publication Data**

US 2003/0112864 A1 Jun. 19, 2003

(51) **Int. Cl.⁷** **H04N 7/12**

(52) **U.S. Cl.** **375/240.17**

(58) **Field of Search** 375/240.17, 240.16;
382/236

A method of interpolation in video coding in which an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, is interpolated to generate values for sub-pixels at fractional horizontal and vertical locations, the method comprising:

a) when values for sub-pixels at half unit horizontal and unit vertical locations, and unit horizontal and half unit vertical locations are required, interpolating such values directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;

b) when values for sub-pixels at half unit horizontal and half unit vertical locations are required, interpolating such values directly using a weighted sum of values for sub-pixels residing at half unit horizontal and unit vertical locations calculated according to step (a); and

c) when values for sub-pixels at quarter unit horizontal and quarter unit vertical locations are required, interpolating such values by taking the average of at least one pair of a first pair of values of a sub-pixel located at a half unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and half unit vertical location and a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a half unit horizontal and half unit vertical location.

(56) **References Cited**

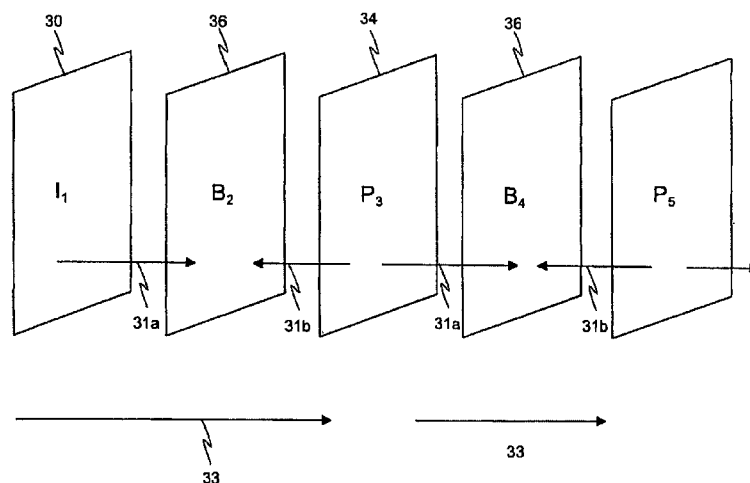
U.S. PATENT DOCUMENTS

4,816,913	A *	3/1989	Harney et al.	375/240.08
5,485,279	A *	1/1996	Yonemitsu et al.	375/240.14
5,521,642	A	5/1996	Park	348/409
5,568,597	A	10/1996	Nakayama et al.	395/132
5,570,436	A	10/1996	Fukushima et al.	382/300
5,594,813	A *	1/1997	Fandrianto et al.	382/236
5,754,240	A *	5/1998	Wilson	375/240.15
5,852,468	A *	12/1998	Okada	348/272
5,901,248	A	5/1999	Fandrianto et al.	382/236
6,104,753	A *	8/2000	Kim et al.	375/240.16
6,219,464	B1	4/2001	Greggain et al.	382/298
6,252,576	B1	6/2001	Nottingham	345/127
6,381,279	B1 *	4/2002	Taubman	375/240.18
6,714,593	B1 *	3/2004	Benzler et al.	375/240.16
2002/0064229	A1	5/2002	Nakaya	375/240.17

FOREIGN PATENT DOCUMENTS

EP 0573290 A2 12/1993

51 Claims, 26 Drawing Sheets



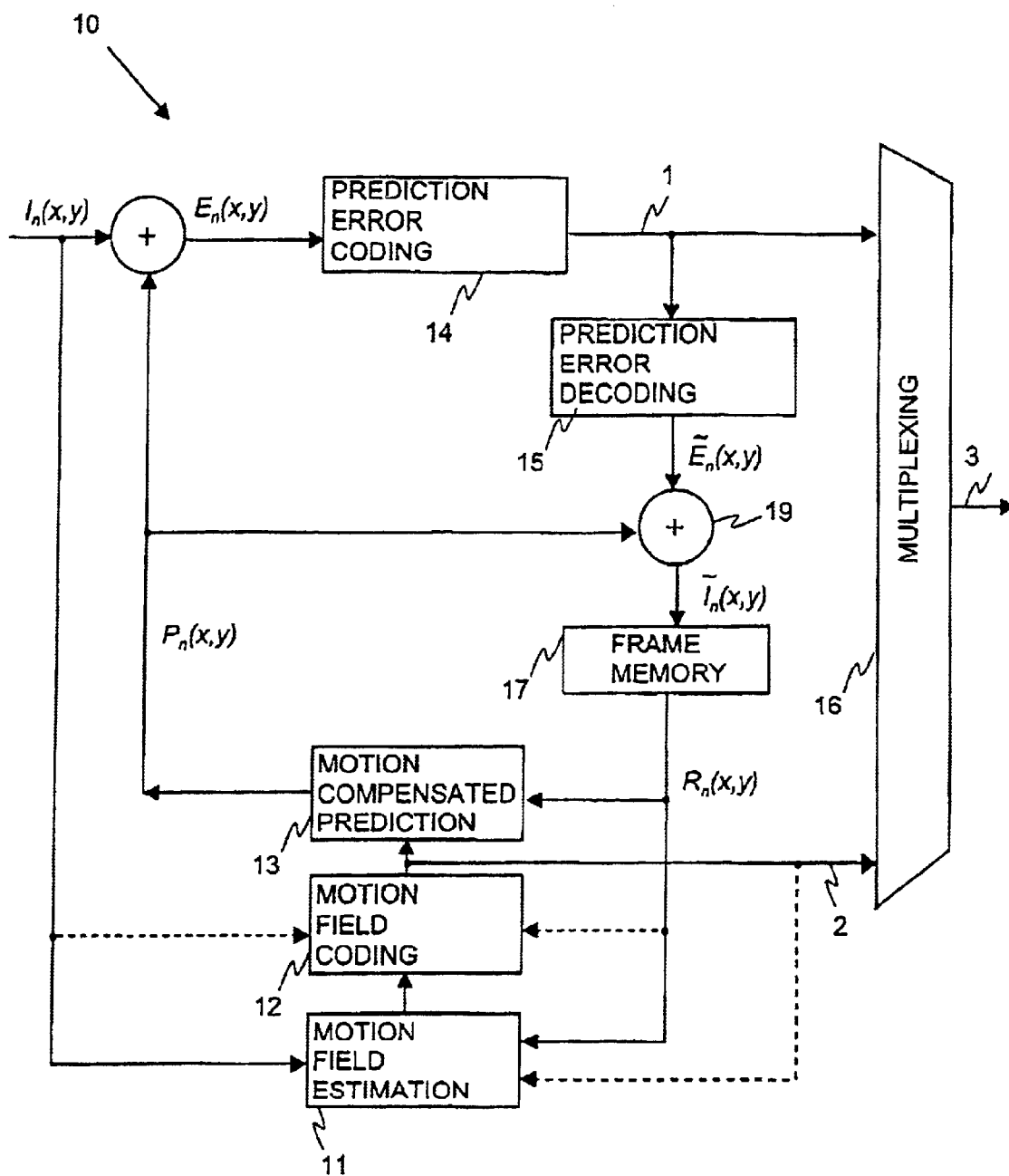
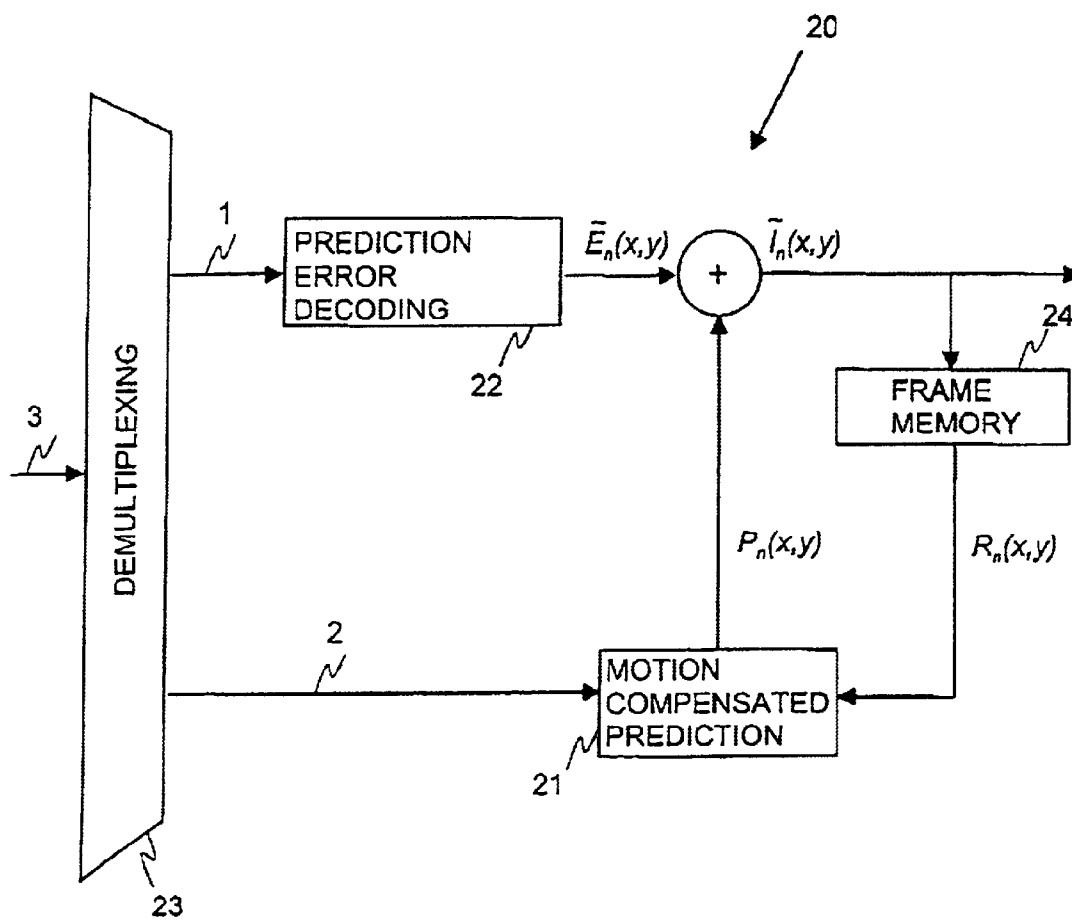


Fig. 1 PRIOR ART



PRIOR ART

Fig. 2

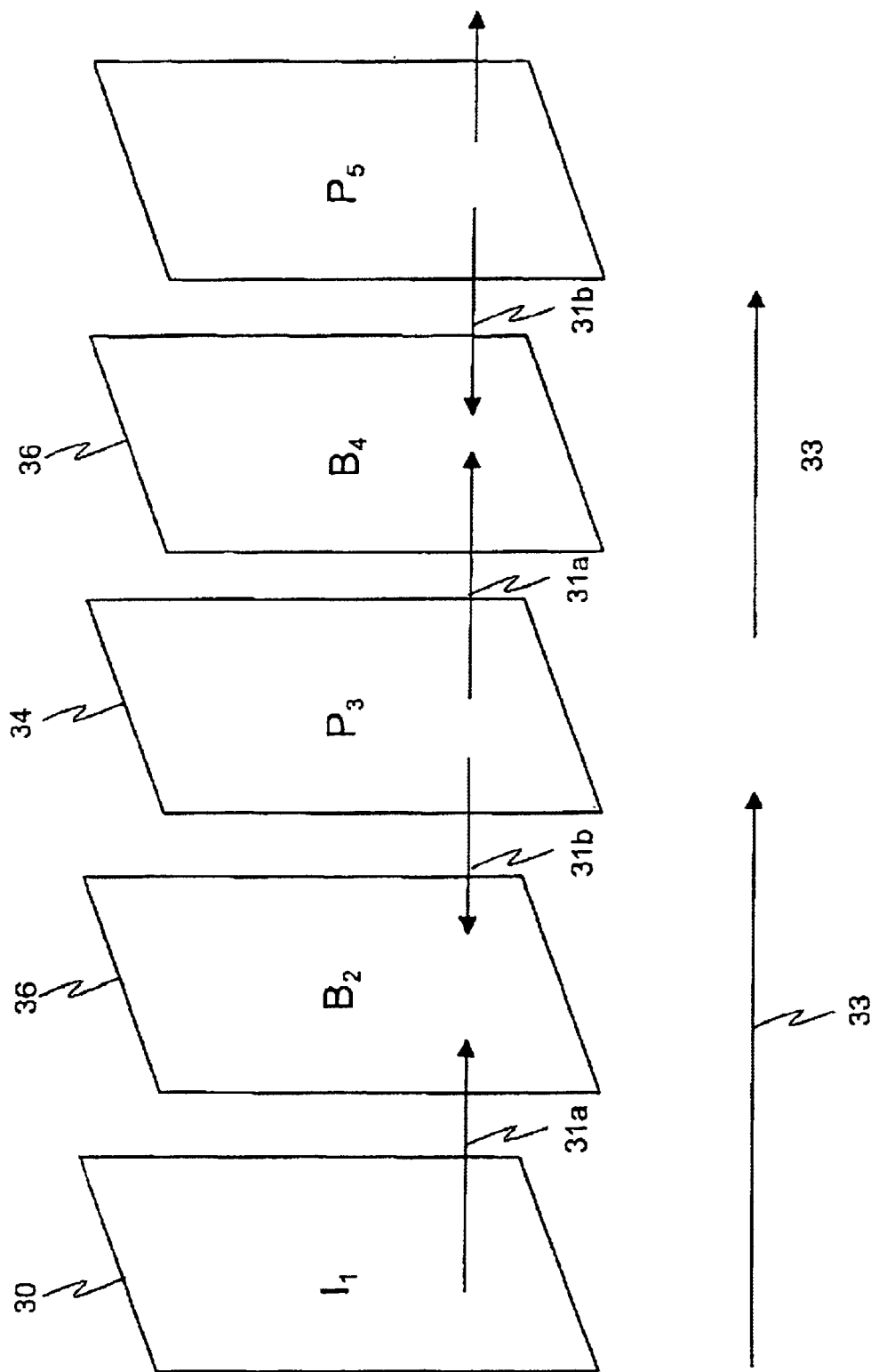


Fig. 3

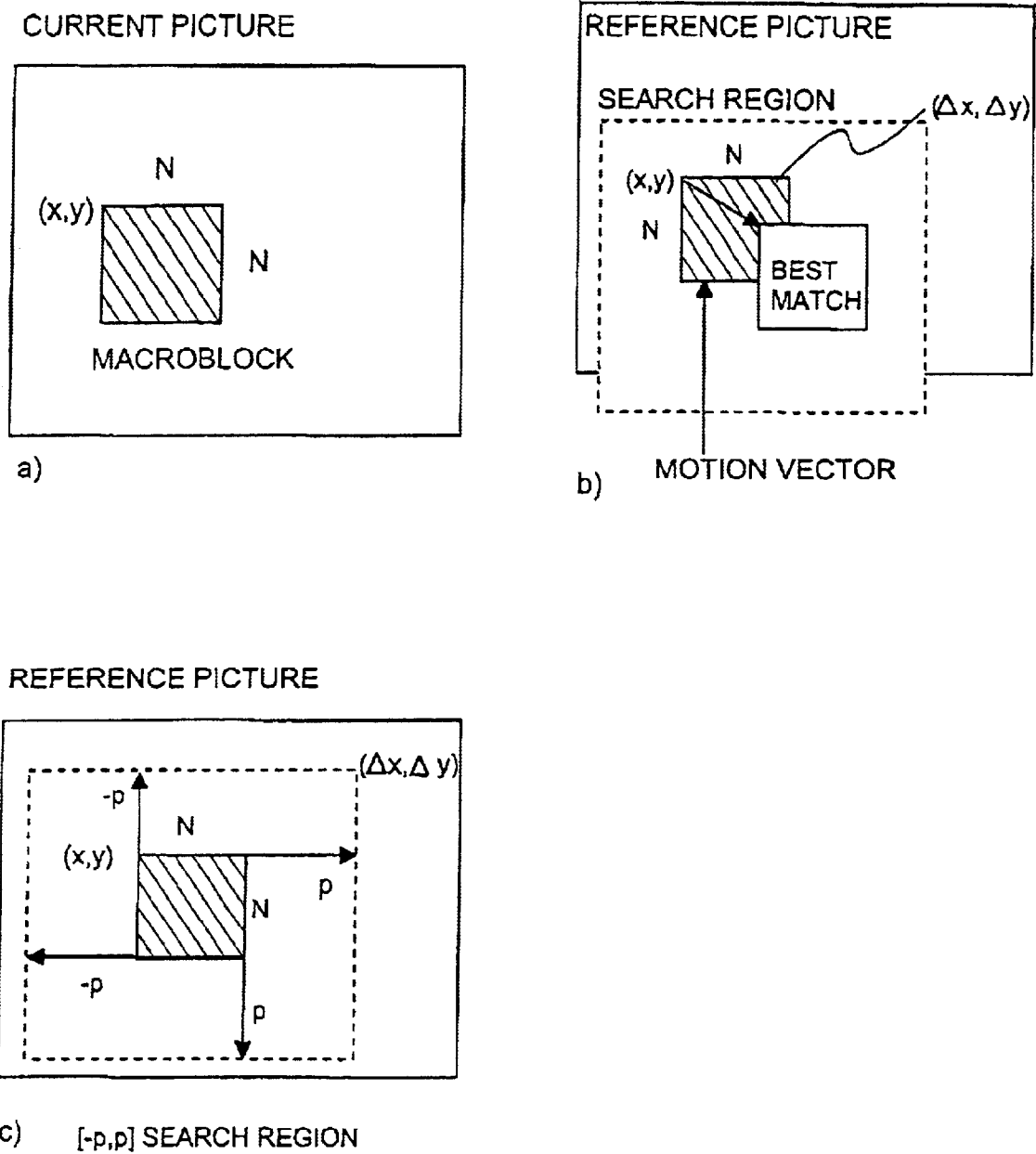


Fig. 4

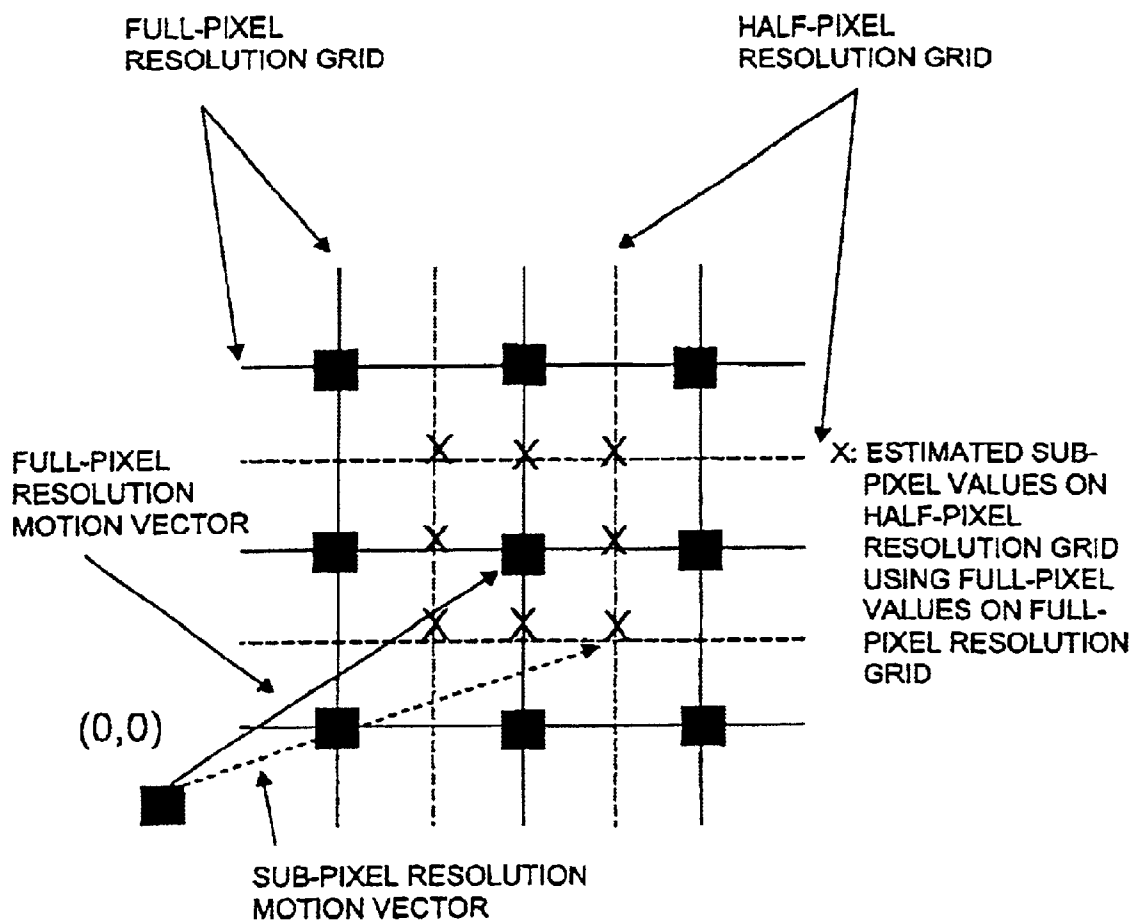


Fig. 5

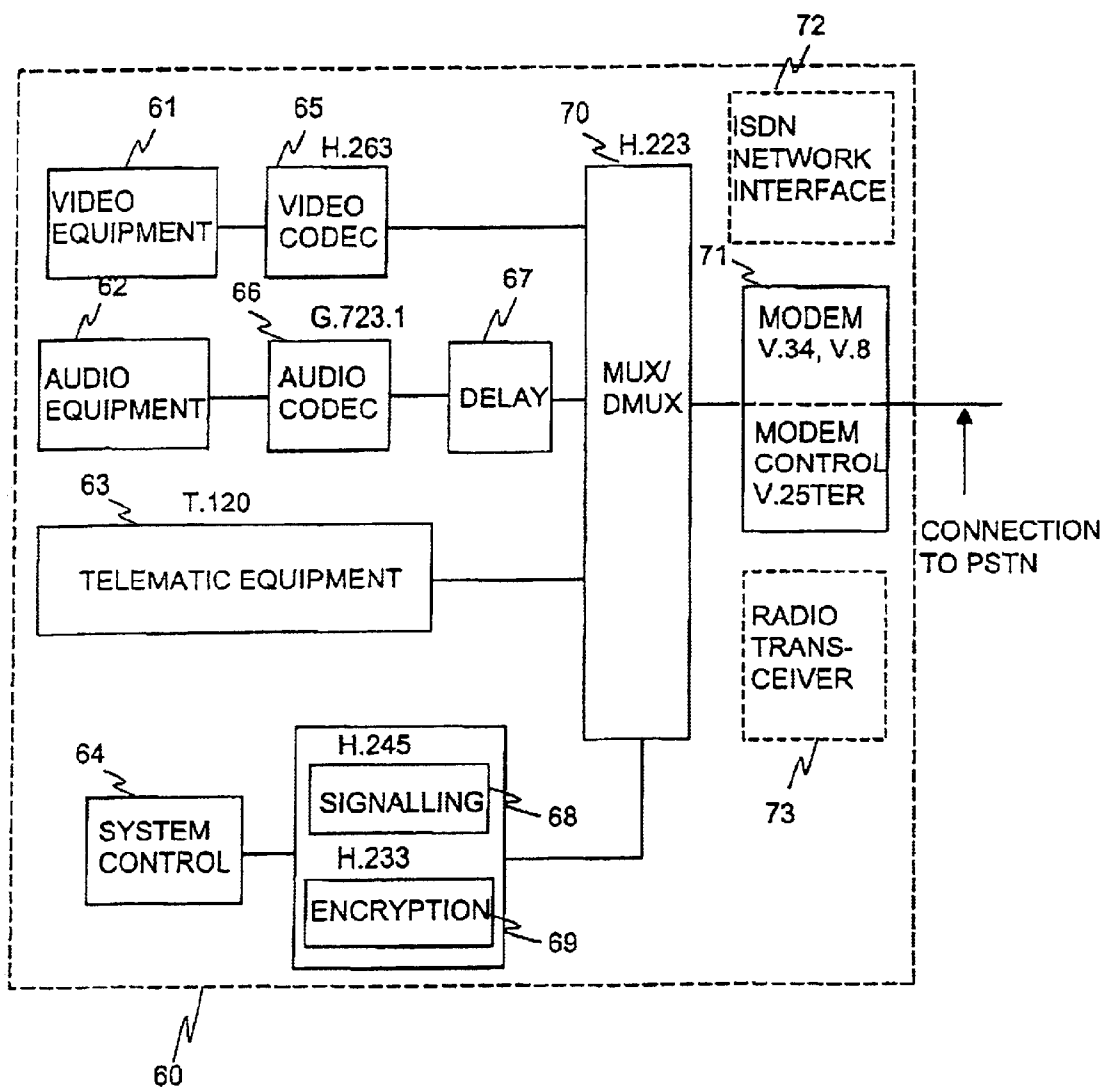


Fig. 6

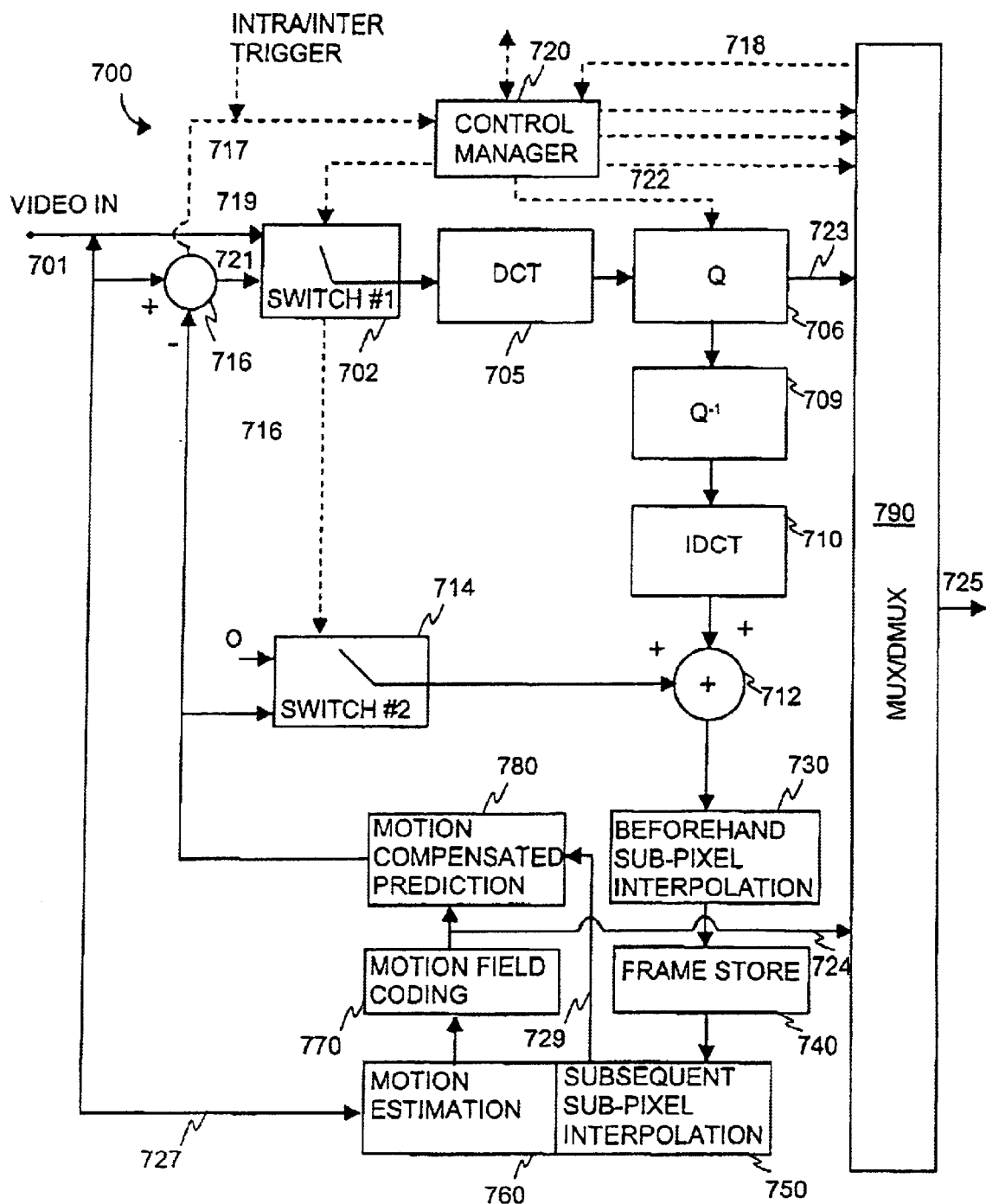


Fig. 7

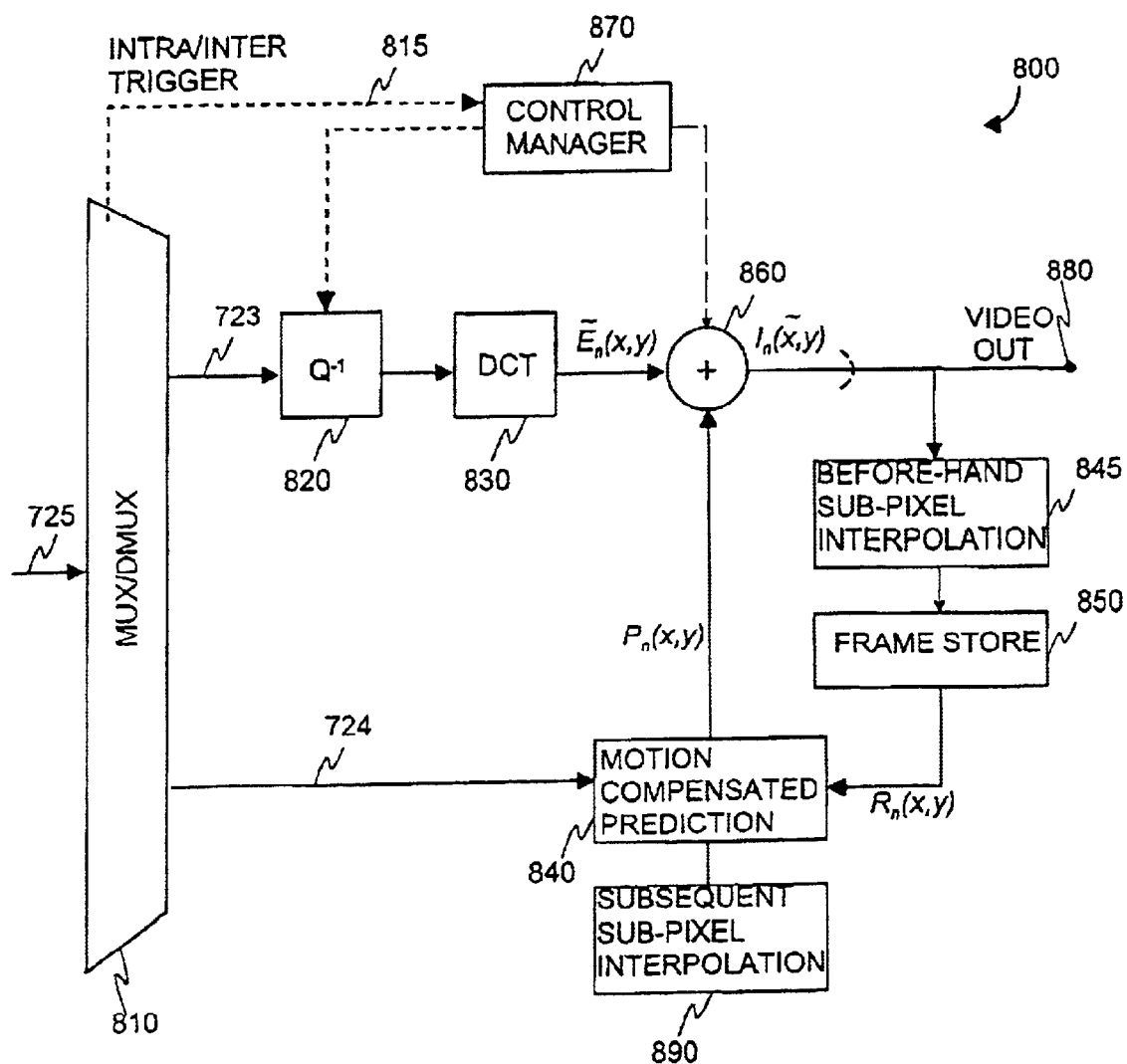


Fig. 8

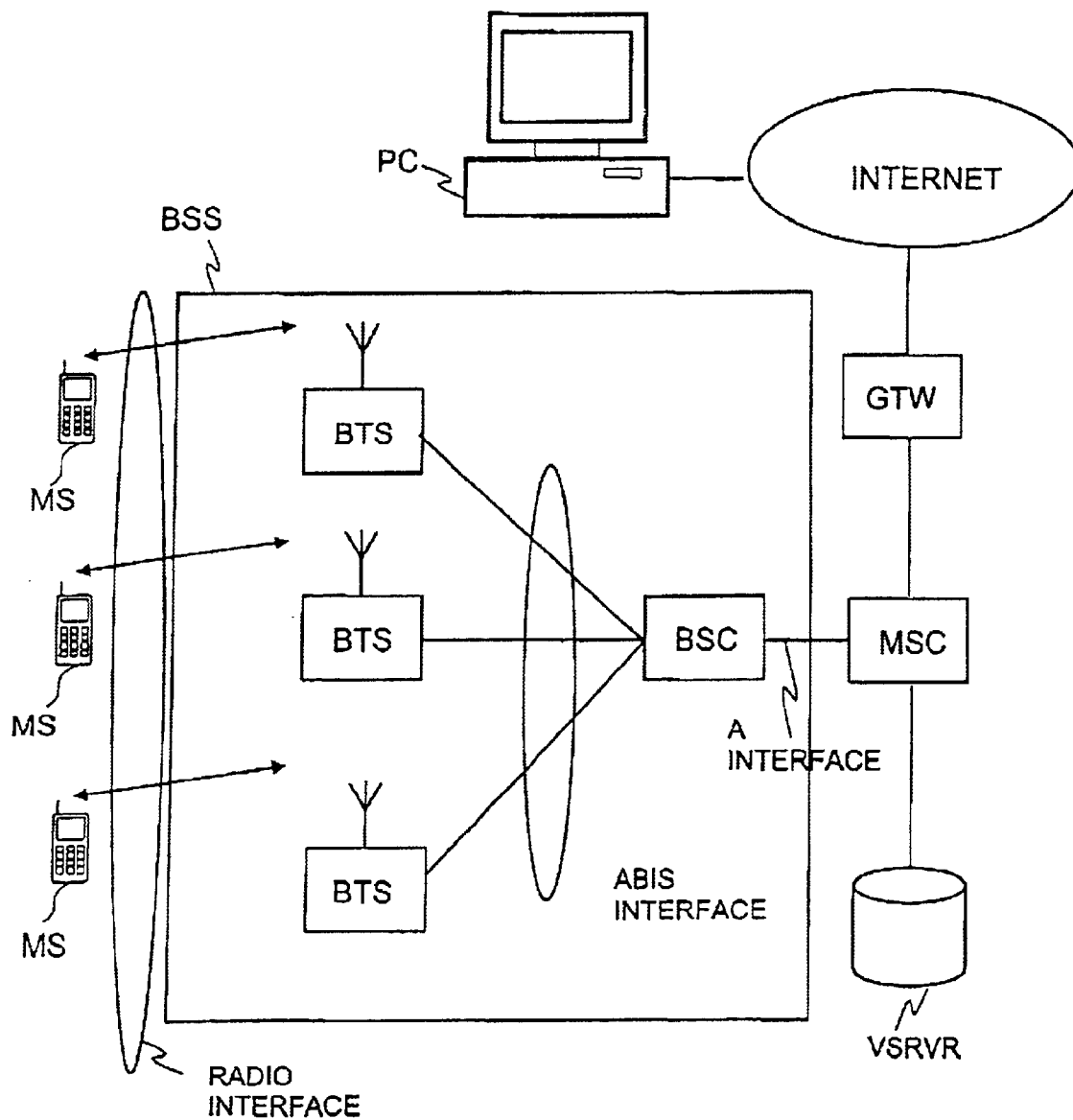


Fig. 11

A	d	b	d	A
e	e	e	e	
c	d	c	d	
e	e	e	f	
A				A

Fig. 12(a)

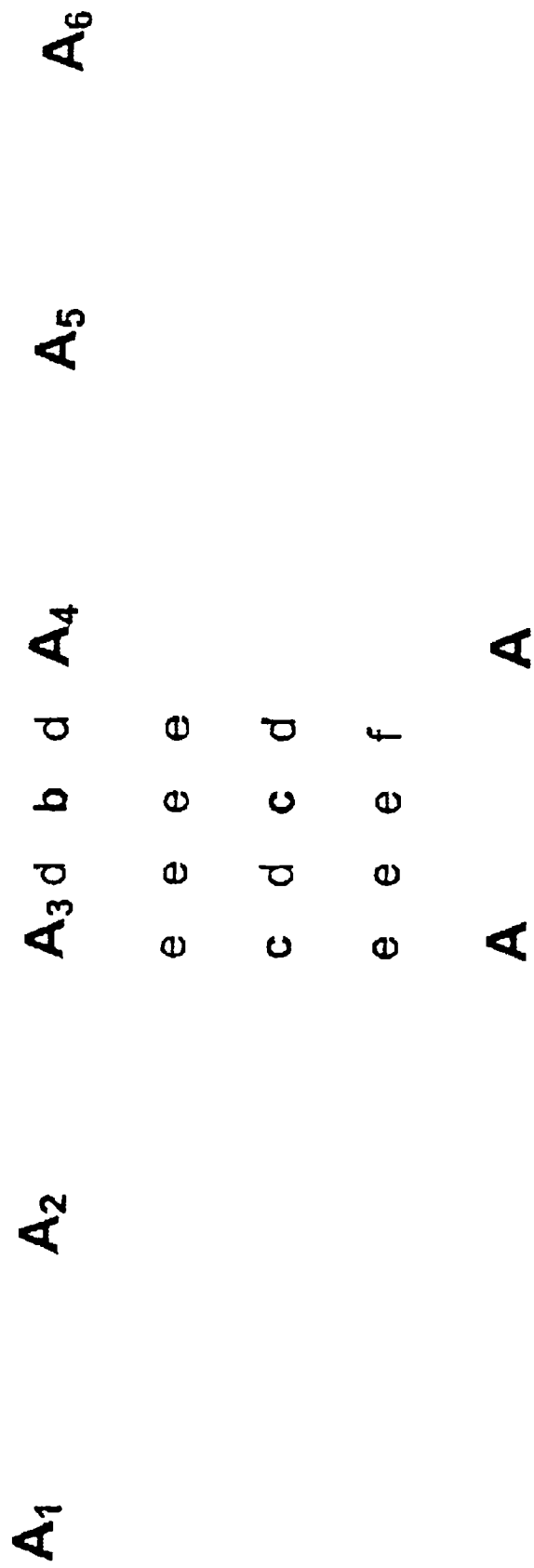


Figure 12(b)

A₁ **b₁**

A₂ **b₂**

A₃	d	b₃	d	A
e	e	e	e	
c	d	c	d	
e	e	e	f	
A₄		b₄		A

A₅ **b₅**

A₆ **b₆**

Figure 12(c)

A	d	b	d	A
e	g	e	g	
c	d	c	d	
e	g	e	f	
A				A

Fig. 13(a)

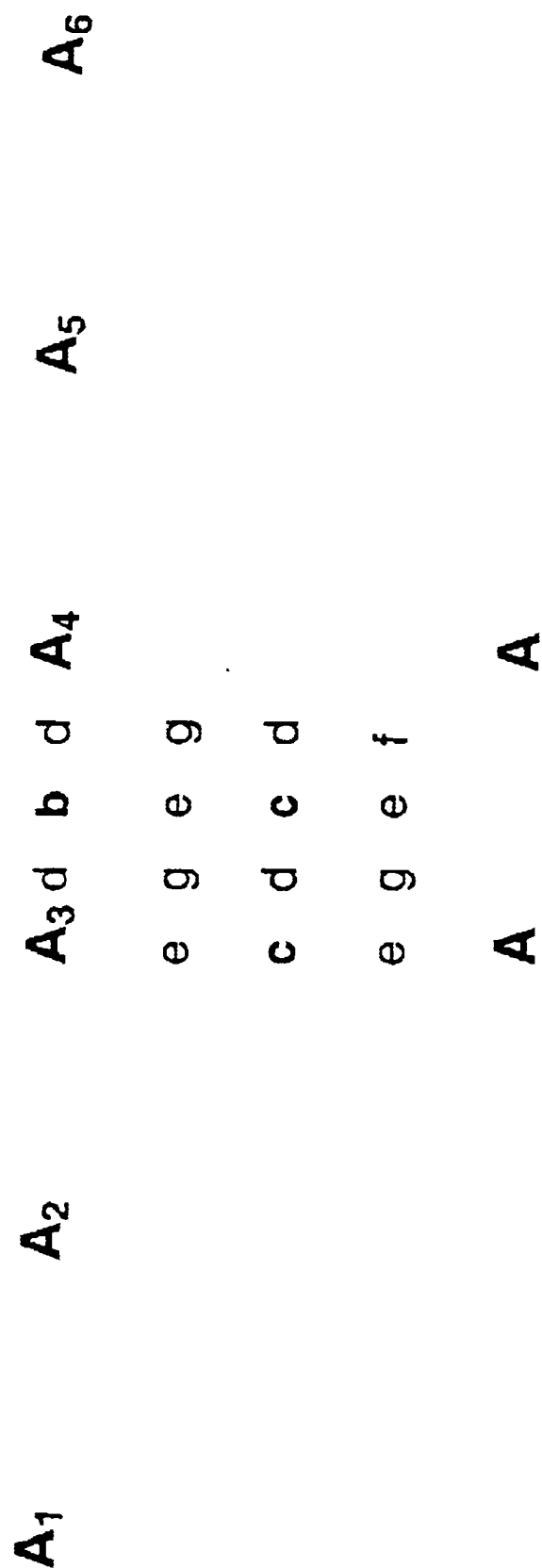


Figure13(b)

A₁ **b₁**

A₂ **b₂**

A₃	d	b₃	d	A
e	g	e	g	
c	d	c	d	
e	g	e	f	
A₄		b₄		A

A₅ **b₅**

A₆ **b₆**

Figure 13(c)

A	d	b	d	A
e	h	f	h	
b	g	c	g	
e	h	f	i	
A				A

Fig. 14(a)

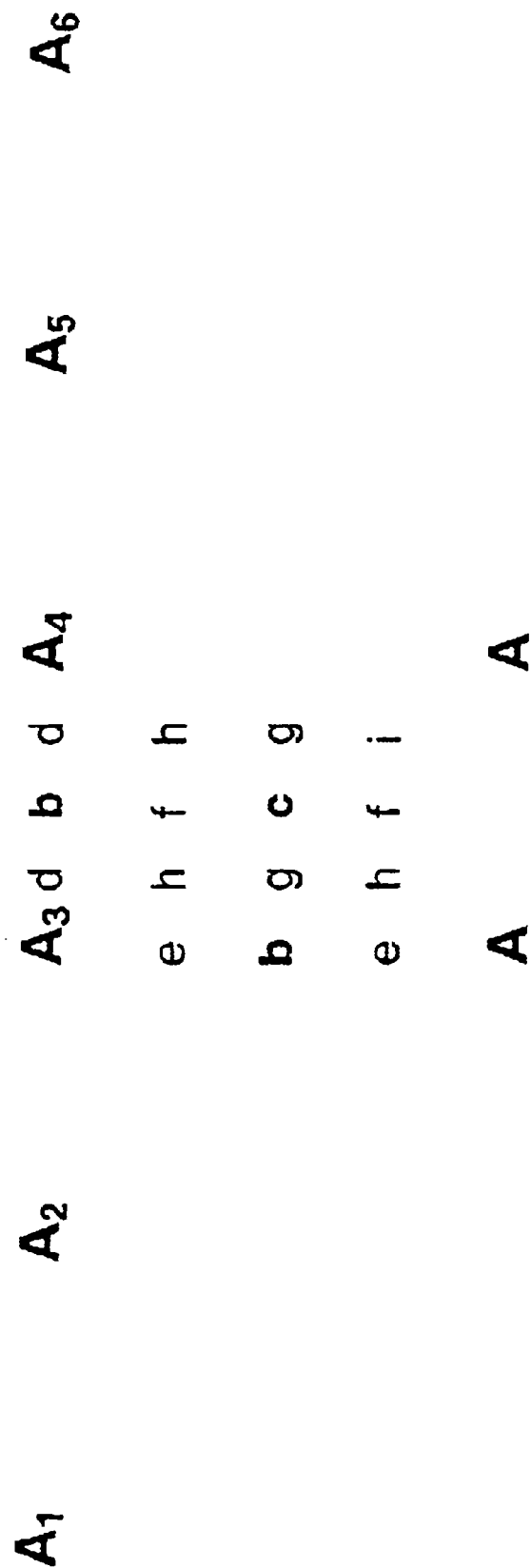


Figure14(b)

A₁

A₂

A₃	d	b	d	A
e	h	f	h	
b	g	c	g	
e	h	f	i	
A₄				A

A₅

A₆

Figure 14(c)

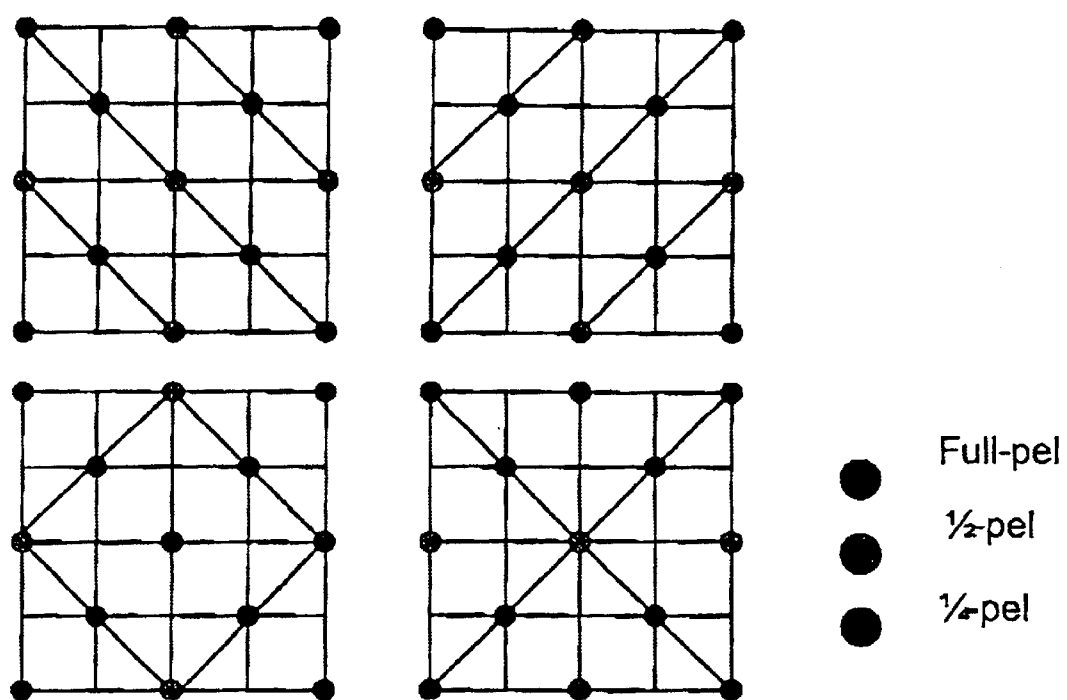


Fig. 15

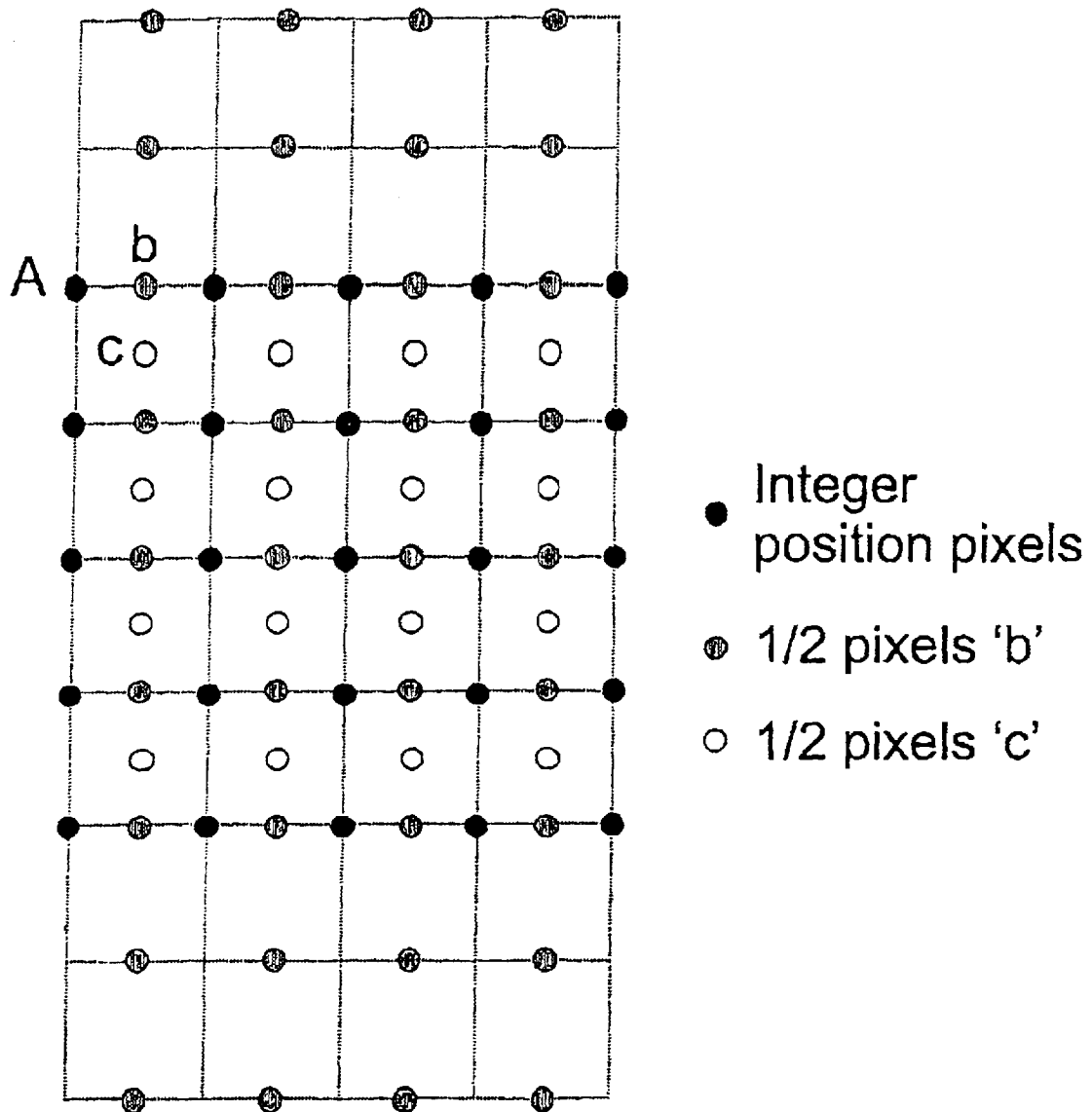


Fig. 16

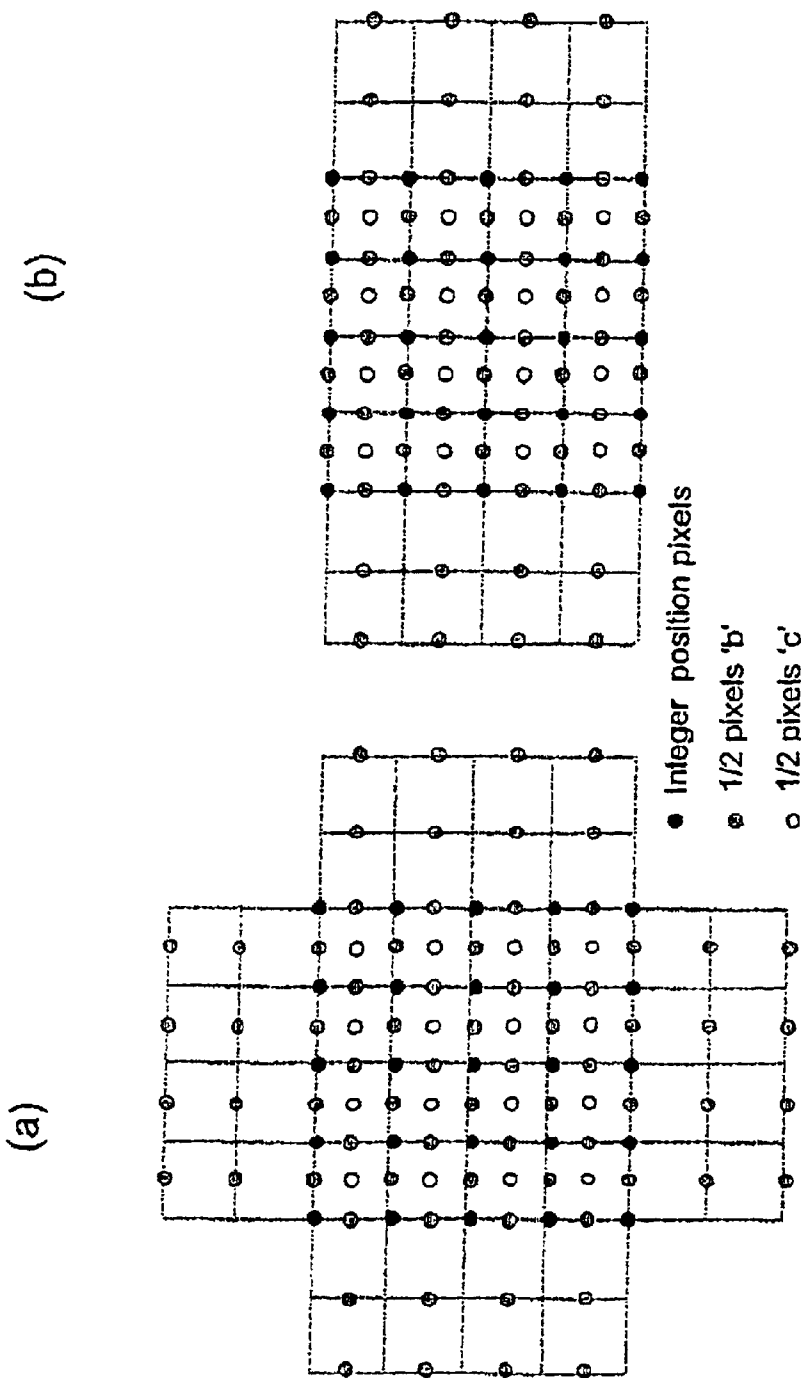


Fig. 17

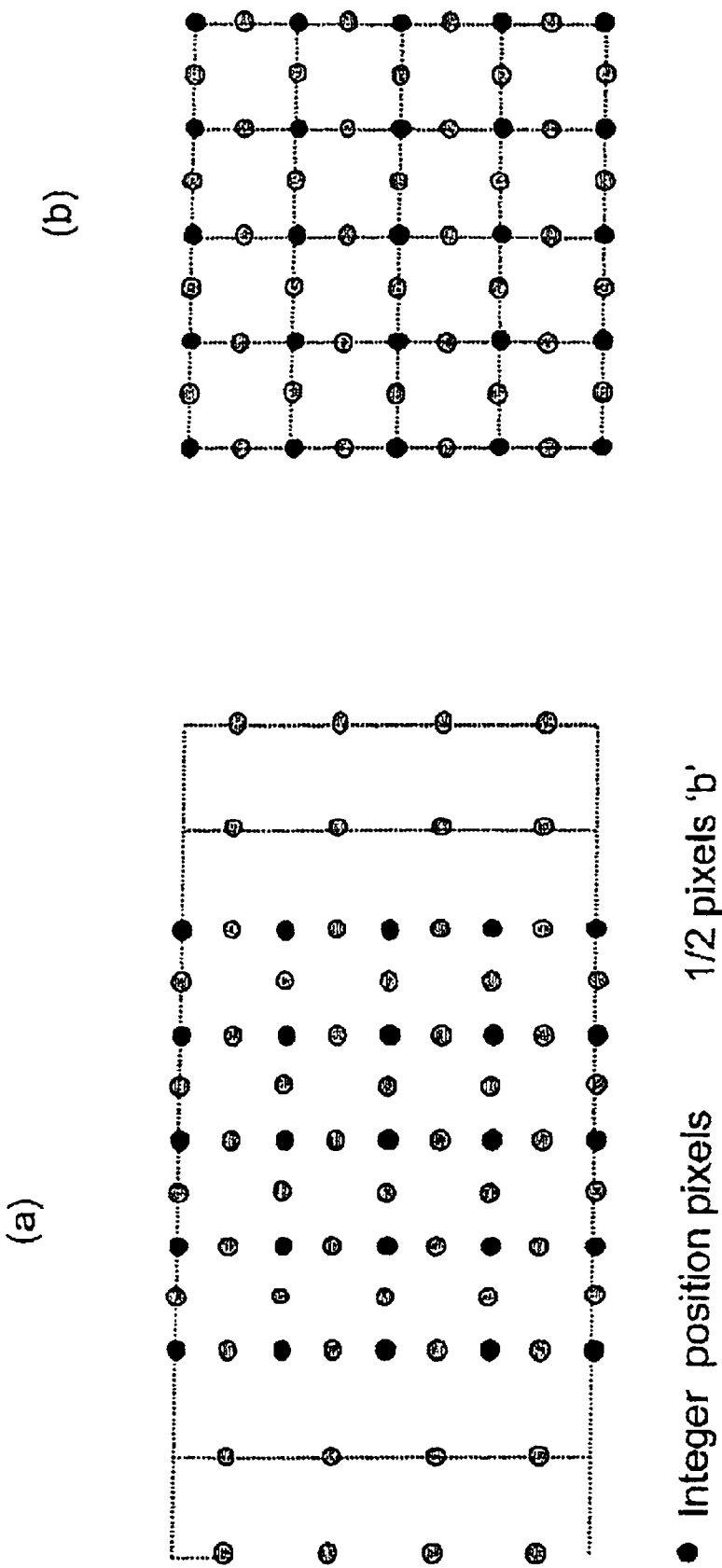


Fig. 18

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

i

Fig. 19

A	d	b ¹	d	b ²	d	b ³	d	A
d	e	d	f	d	f	d	e	
b ¹	d	c ¹¹	d	c ¹²	d	c ¹³	d	
d	f	d	g	d	g	d	f	
b ²	d	c ²¹	d	c ²²	d	c ²³	d	
d	f	d	g	d	g	d	e	
b ³	d	c ³¹	d	c ³²	d	c ³³	d	
d	e	d	f	d	f	d	e	
A								

Figure 20

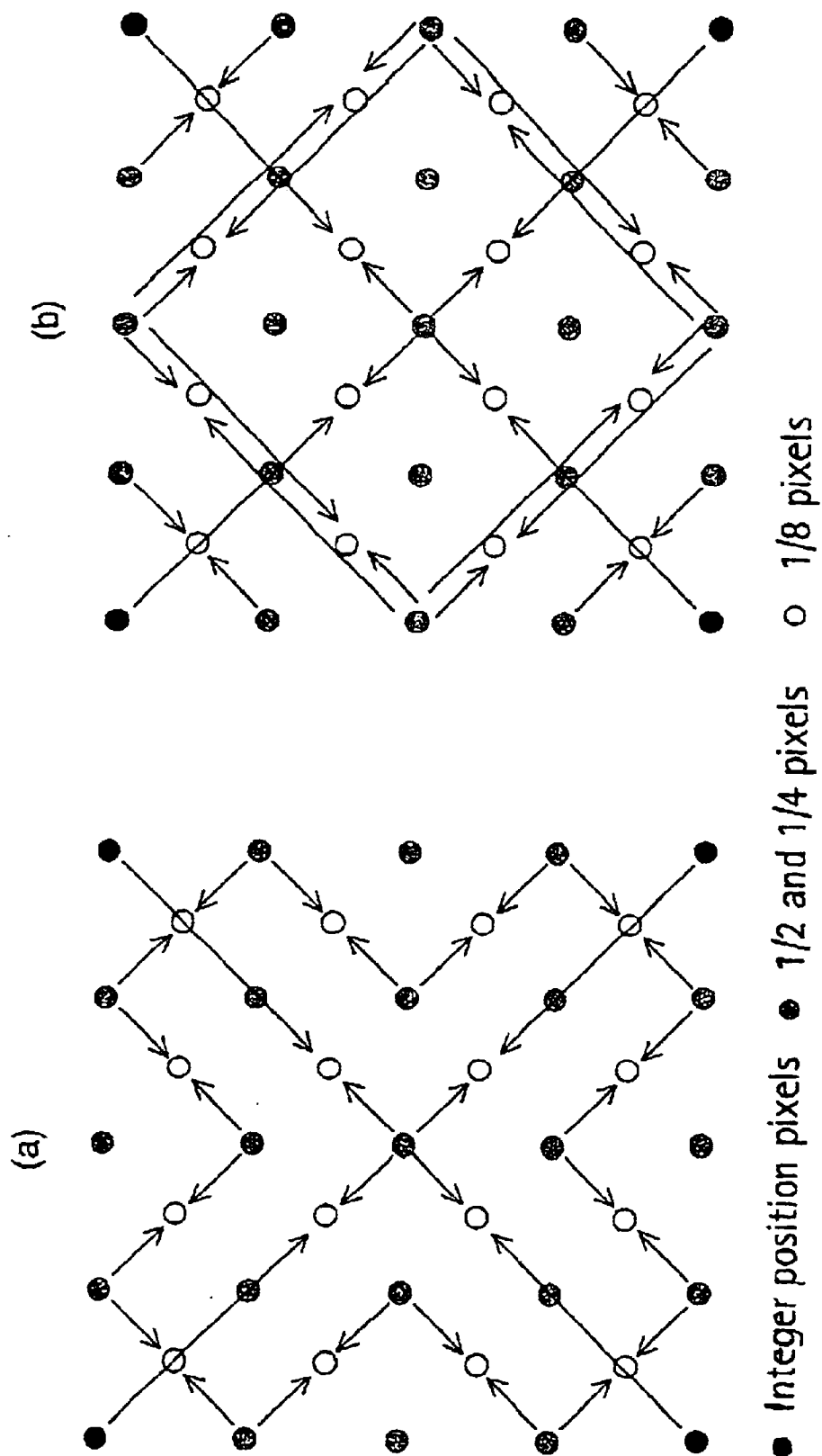


Fig. 21

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

Fig. 22

1

METHOD FOR SUB-PIXEL VALUE INTERPOLATION

The present invention relates to a method for sub-pixel value interpolation in the encoding and decoding of data. It relates particularly, but not exclusively, to encoding and decoding of digital video.

BACKGROUND OF THE INVENTION

Digital video sequences, like ordinary motion pictures recorded on film, comprise a sequence of still images, the illusion of motion being created by displaying the images one after the other at a relatively fast frame rate, typically 15 to 30 frames per second. Because of the relatively fast frame rate, images in consecutive frames tend to be quite similar and thus contain a considerable amount of redundant information. For example, a typical scene may comprise some stationary elements, such as background scenery, and some moving areas, which may take many different forms, for example the face of a newsreader, moving traffic and so on. Alternatively, the camera recording the scene may itself be moving, in which case all elements of the image have the same kind of motion. In many cases, this means that the overall change between one video frame and the next is rather small. Of course, this depends on the nature of the movement. For example, the faster the movement, the greater the change from one frame to the next. Similarly, if a scene contains a number of moving elements, the change from one frame to the next is likely to be greater than in a scene where only one element is moving.

It should be appreciated that each frame of a raw, that is uncompressed, digital video sequence comprises a very large amount of image information. Each frame of an uncompressed digital video sequence is formed from an array of image pixels. For example, in a commonly used digital video format, known as the Quarter Common Interchange Format (QCIF), a frame comprises an array of 176x144 pixels, in which case each frame has 25,344 pixels. In turn, each pixel is represented by a certain number of bits, which carry information about the luminance and/or colour content of the region of the image corresponding to the pixel. Commonly, a so-called YUV colour model is used to represent the luminance and chrominance content of the image. The luminance, or Y, component represents the intensity (brightness) of the image, while the colour content of the image is represented by two chrominance components, labelled U and V.

Colour models based on a luminance/chrominance representation of image content provide certain advantages compared with colour models that are based on a representation involving primary colours (that is Red, Green and Blue, RGB). The human visual system is more sensitive to intensity variations than it is to colour variations; YUV colour models exploit this property by using a lower spatial resolution for the chrominance components (U, V) than for the luminance component (Y). In this way the amount of information needed to code the colour information in an image can be reduced with an acceptable reduction in image quality.

The lower spatial resolution of the chrominance components is usually attained by sub-sampling. Typically, a block of 16x16 image pixels is represented by one block of 16x16 pixels comprising luminance information and the corresponding chrominance components are each represented by one block of 8x8 pixels representing an area of the image equivalent to that of the 16x16 pixels of the luminance

2

component. The chrominance components are thus spatially sub-sampled by a factor of 2 in the x and y directions. The resulting assembly of one 16x16 pixel luminance block and two 8x8 pixel chrominance blocks is commonly referred to as a YUV macroblock, or macroblock, for short.

A QCIF image comprises 11x9 macroblocks. If the luminance blocks and chrominance blocks are represented with 8 bit resolution (that is by numbers in the range 0 to 255), the total number of bits required per macroblock is $(16 \times 16 \times 8) + 2 \times (8 \times 8 \times 8) = 3072$ bits. The number of bits needed to represent a video frame in QCIF format is thus $99 \times 3072 = 304,128$ bits. This means that the amount of data required to transmit/record/display a video sequence in QCIF format, represented using a YUV colour model, at a rate of 30 frames per second, is more than 9 Mbps (million bits per second). This is an extremely high data rate and is impractical for use in video recording, transmission and display applications because of the very large storage capacity, transmission channel capacity and hardware performance required.

If video data is to be transmitted in real-time over a fixed line network such as an ISDN (Integrated Services Digital Network) or a conventional PSTN (Public Service Telephone Network), the available data transmission bandwidth is typically of the order of 64 kbits/s. In mobile videotelephony, where transmission takes place at least in part over a radio communications link, the available bandwidth can be as low as 20 kbits/s. This means that a significant reduction in the amount of information used to represent video data must be achieved in order to enable transmission of digital video sequences over low bandwidth communication networks. For this reason video compression techniques have been developed which reduce the amount of information transmitted while retaining an acceptable image quality.

Video compression methods are based on reducing the redundant and perceptually irrelevant parts of video sequences. The redundancy in video sequences can be categorised into spatial, temporal and spectral redundancy. 'Spatial redundancy' is the term used to describe the correlation between neighbouring pixels within a frame. The term 'temporal redundancy' expresses the fact that the objects appearing in one frame of a sequence are likely to appear in subsequent frames, while 'spectral redundancy' refers to the correlation between different colour components of the same image.

Sufficiently efficient compression cannot usually be achieved by simply reducing the various forms of redundancy in a given sequence of images. Thus, most current video encoders also reduce the quality of those parts of the video sequence which are subjectively the least important. In addition, the redundancy of the compressed video bit-stream is itself reduced by means of efficient loss-less encoding. Typically, this is achieved using a technique known as 'variable length coding' (VLC).

Modern video compression standards, such as ITU-T recommendations H.261, H.263(+)(++), H.26L and the Motion Picture Experts Group recommendation MPEG-4 make use of 'motion compensated temporal prediction'. This is a form of temporal redundancy reduction in which the content of some (often many) frames in a video sequence is 'predicted' from other frames in the sequence by tracing the motion of objects or regions of an image between frames.

Compressed images which do not make use of temporal redundancy reduction are usually called INTRA-coded or

I-frames, whereas temporally predicted images are called INTER-coded or P-frames. In the case of INTER frames, the predicted (motion-compensated) image is rarely precise enough to represent the image content with sufficient quality, and therefore a spatially compressed prediction error (PE) frame is also associated with each INTER frame. Many video compression schemes can also make use of bi-directionally predicted frames, which are commonly referred to as B-pictures or B-frames. B-pictures are inserted between reference or so-called 'anchor' picture pairs (I or P frames) and are predicted from either one or both of the anchor pictures. B-pictures are not themselves used as anchor pictures, that is no other frames are predicted from them, and therefore, they can be discarded from the video sequence without causing deterioration in the quality of future pictures.

The different types of frame that occur in a typical compressed video sequence are illustrated in FIG. 3 of the accompanying drawings. As can be seen from the figure, the sequence starts with an INTRA or I frame 30. In FIG. 3, arrows 33 denote the 'forward' prediction process by which P-frames (labelled 34) are formed. The bidirectional prediction process by which B-frames (36) are formed is denoted by arrows 31a and 31b, respectively.

A schematic diagram of an example video coding system using motion compensated prediction is shown in FIGS. 1 and 2. FIG. 1 illustrates an encoder 10 employing motion compensation and FIG. 2 illustrates a corresponding decoder 20. The encoder 10 shown in FIG. 1 comprises a Motion Field Estimation block 11, a Motion Field Coding block 12, a Motion Compensated Prediction block 13, a Prediction Error Coding block 14, a Prediction Error Decoding block 15, a Multiplexing block 16, a Frame Memory 17, and an adder 19. The decoder 20 comprises a Motion Compensated Prediction block 21, a Prediction Error Decoding block 22, a Demultiplexing block 23 and a Frame Memory 24.

The operating principle of video coders using motion compensation is to minimise the amount of information in a prediction error frame $E_n(x,y)$, which is the difference between a current frame $I_n(x,y)$ being coded and a prediction frame $P_n(x,y)$. The prediction error frame is thus:

$$E_n(x,y) = I_n(x,y) - P_n(x,y). \quad (1)$$

The prediction frame $P_n(x,y)$ is built using pixel values of a reference frame $R_n(x,y)$, which is generally one of the previously coded and transmitted frames, for example the frame immediately preceding the current frame and is available from the Frame Memory 17 of the encoder 10. More specifically, the prediction frame $P_n(x,y)$ is constructed by finding so-called 'prediction pixels' in the reference frame $R_n(x,y)$ which correspond substantially with pixels in the current frame. Motion information, describing the relationship (e.g. relative location, rotation, scale etc.) between pixels in the current frame and their corresponding prediction pixels in the reference frame is derived and the prediction frame is constructed by moving the prediction pixels according to the motion information. In this way, the prediction frame is constructed as an approximate representation of the current frame, using pixel values in the reference frame. The prediction error frame referred to above therefore represents the difference between the approximate representation of the current frame provided by the prediction frame and the current frame itself. The basic advantage provided by video encoders that use motion compensated prediction arises from the fact that a comparatively compact description of the current frame can be obtained by representing it

in terms of the motion information required to form its prediction together with the associated prediction error information in the prediction error frame.

However, due to the very large number of pixels in a frame, it is generally not efficient to transmit separate motion information for each pixel to the decoder. Instead, in most video coding schemes, the current frame is divided into larger image segments S_k and motion information relating to the segments is transmitted to the decoder. For example, motion information is typically provided for each macroblock of a frame and the same motion information is then used for all pixels within the macroblock. In some video coding standards, such as H.26L, a macroblock can be divided into smaller blocks, each smaller block being provided with its own motion information.

The motion information usually takes the form of motion vectors $[\Delta x(x,y), \Delta y(x,y)]$. The pair of numbers $\Delta x(x,y)$ and $\Delta y(x,y)$ represents the horizontal and vertical displacements of a pixel at location (x,y) in the current frame $I_n(x,y)$ with respect to a pixel in the reference frame $R_n(x,y)$. The motion vectors $[\Delta x(x,y), \Delta y(x,y)]$ are calculated in the Motion Field Estimation block 11 and the set of motion vectors of the current frame $[\Delta x(\bullet), \Delta y(\bullet)]$ is referred to as the motion vector field.

Typically, the location of a macroblock in a current video frame is specified by the (x,y) co-ordinate of its upper left-hand corner. Thus, in a video coding scheme in which motion information is associated with each macroblock of a frame, each motion vector describes the horizontal and vertical displacement $\Delta x(x,y)$ and $\Delta y(x,y)$ of a pixel representing the upper left-hand corner of a macroblock in the current frame $I_n(x,y)$ with respect to a pixel in the upper left-hand corner of a substantially corresponding block of prediction pixels in the reference frame $R_n(x,y)$ (as shown in FIG. 4b).

Motion estimation is a computationally intensive task. Given a reference frame $R_n(x,y)$ and, for example, a square macroblock comprising $N \times N$ pixels in a current frame (as shown in FIG. 4a), the objective of motion estimation is to find an $N \times N$ pixel block in the reference frame that matches the characteristics of the macroblock in the current picture according to some criterion. This criterion can be, for example, a sum of absolute differences (SAD) between the pixels of the macroblock in the current frame and the block of pixels in the reference frame with which it is compared. This process is known generally as 'block matching'. It should be noted that, in general, the geometry of the block to be matched and that in the reference frame do not have to be the same, as real-world objects can undergo scale changes, as well as rotation and warping. However, in current international video coding standards, only a translational motion model is used (see below) and thus fixed rectangular geometry is sufficient.

Ideally, in order to achieve the best chance of finding a match, the whole of the reference frame should be searched. However, this is impractical as it imposes too high a computational burden on the video encoder. Instead, the search region is restricted to region $[-p,p]$ around the original location of the macroblock in the current frame, as shown in FIG. 4c.

In order to reduce the amount of motion information to be transmitted from the encoder 10 to the decoder 20, the motion vector field is coded in the Motion Field Coding block 12 of the encoder 10, by representing it with a motion model. In this process, the motion vectors of image segments are re-expressed using certain predetermined functions or, in other words, the motion vector field is repre-

5

sented with a model. Almost all currently used motion vector field models are additive motion models, complying with the following general formula:

$$\Delta x(x, y) = \sum_{i=0}^{N-1} a_i f_i(x, y) \quad (2)$$

$$\Delta y(x, y) = \sum_{i=0}^{M-1} b_i g_i(x, y) \quad (3)$$

where coefficients a_i and b_i are called motion coefficients. The motion coefficients are transmitted to the decoder **20** (information stream **2** in FIGS. 1 and 2). Functions f_i and g_i are called motion field basis functions, and are known both to the encoder and decoder. An approximate motion vector field ($\Delta x(x, y), \Delta y(x, y)$) can be constructed using the coefficients and the basis functions. As the basis functions are known to (that is stored in) both the encoder **10** and the decoder **20**, only the motion coefficients need to be transmitted to the encoder, thus reducing the amount of information required to represent the motion information of the frame.

The simplest motion model is the translational motion model which requires only two coefficients to describe the motion vectors of each segment. The values of motion vectors are given by:

$$\begin{aligned} \Delta x(x, y) &= a_0 \\ \Delta y(x, y) &= b_0 \end{aligned} \quad (4)$$

This model is widely used in various international standards (ISO MPEG-1, MPEG-2, MPEG-4, ITU-T Recommendations H.261 and H.263) to describe the motion of 16×16 and 8×8 pixel blocks. Systems which use a translational motion model typically perform motion estimation at full pixel resolution or some integer fraction of full pixel resolution, for example at half or one quarter pixel resolution.

The prediction frame $P_n(x, y)$ is constructed in the Motion Compensated Prediction block **13** in the encoder **10**, and is given by:

$$P_n(x, y) = R_n[x + \tilde{\Delta}x(x, y), y + \tilde{\Delta}y(x, y)] \quad (5)$$

In the Prediction Error Coding block **14**, the prediction error frame $E_n(x, y)$ is typically compressed by representing it as a finite series (transform) of some 2-dimensional functions. For example, a 2-dimensional Discrete Cosine Transform (DCT) can be used. The transform coefficients are quantised and entropy (for example Huffman) coded before they are transmitted to the decoder (information stream **1** in FIGS. 1 and 2). Because of the error introduced by quantisation, this operation usually produces some degradation (loss of information) in the prediction error frame $E_n(x, y)$. To compensate for this degradation, the encoder **10** also comprises a Prediction Error Decoding block **15**, where a decoded prediction error frame $\tilde{E}_n(x, y)$ is constructed using the transform coefficients. This locally decoded prediction error frame is added to the prediction frame $P_n(x, y)$ in the adder **19** and the resulting decoded current frame $\tilde{I}_n(x, y)$ is stored in the Frame Memory **17** for further use as the next reference frame $R_{n+1}(x, y)$.

The information stream **2** carrying information about the motion vectors is combined with information about the prediction error in multiplexer **16** and an information stream **3** containing typically at least those two types of information is sent to the decoder **20**.

6

The operation of a corresponding video decoder **20** will now be described.

The Frame Memory **24** of the decoder **20** stores a previously reconstructed reference frame $R_n(x, y)$. The prediction frame $P_n(x, y)$ is constructed in the Motion Compensated Prediction block **21** of the decoder **20** according to equation 5, using received motion coefficient information and pixel values of the previously reconstructed reference frame $R_n(x, y)$. The transmitted transform coefficients of the prediction error frame $E_n(x, y)$ are used in the Prediction Error Decoding block **22** to construct the decoded prediction error frame $\tilde{E}_n(x, y)$. The pixels of the decoded current frame $\tilde{I}_n(x, y)$ are then reconstructed by adding the prediction frame $P_n(x, y)$ and the decoded prediction error frame $\tilde{E}_n(x, y)$:

$$\tilde{I}_n(x, y) = P_n(x, y) + \tilde{E}_n(x, y) = R_n[x + \tilde{\Delta}x(x, y), y + \tilde{\Delta}y(x, y)] + \tilde{E}_n(x, y). \quad (6)$$

This decoded current frame may be stored in the Frame Memory **24** as the next reference frame $R_{n+1}(x, y)$.

In the description of motion compensated encoding and decoding of digital video presented above, the motion vector $[\Delta x(x, y), \Delta y(x, y)]$ describing the motion of a macroblock in the current frame with respect to the reference frame $R_n(x, y)$ can point to any of the pixels in the reference frame. This means that motion between frames of a digital video sequence can only be represented at a resolution which is determined by the image pixels in the frame (so-called full pixel resolution). Real motion, however, has arbitrary precision, and thus the system described above can only provide approximate modelling of the motion between successive frames of a digital video sequence. Typically, modelling of motion between video frames with full pixel resolution is not sufficiently accurate to allow efficient minimisation of the prediction error (PE) information associated with each macroblock/frame. Therefore, to enable more accurate modelling of real motion and to help reduce the amount of PE information that must be transmitted from encoder to decoder, many video coding standards, such as H.263(++) and H.26L, allow motion vectors to point 'in between' image pixels. In other words, the motion vectors can have 'sub-pixel' resolution. Allowing motion vectors to have sub-pixel resolution adds to the complexity of the encoding and decoding operations that must be performed, so it is still advantageous to limit the degree of spatial resolution a motion vector may have. Thus, video coding standards, such as those previously mentioned, typically only allow motion vectors to have full-, half- or quarter-pixel resolution.

Motion estimation with sub-pixel resolution is usually performed as a two-stage process, as illustrated in FIG. 5, for a video coding scheme which allows motion vectors to have full- or half-pixel resolution. In the first step, a motion vector having full-pixel resolution is determined using any appropriate motion estimation scheme, such as the block-matching process described in the foregoing. The resulting motion vector, having full-pixel resolution is shown in FIG. 5.

In the second stage, the motion vector determined in the first stage is refined to obtain the desired half-pixel resolution. In the example illustrated in FIG. 5, this is done by forming eight new search blocks of 16×16 pixels, the location of the top-left corner of each block being marked with an X in FIG. 5. These locations are denoted as $[\Delta x + m/2, \Delta y + n/2]$, where m and n can take the values -1, 0 and +1, but cannot be zero at the same time. As only the pixel values of original image pixels are known, the values (for example luminance and/or chrominance values) of the

sub-pixels residing at half-pixel locations must be estimated for each of the eight new search blocks, using some form of interpolation scheme.

Having interpolated the values of the sub-pixels at half-pixel resolution, each of the eight search blocks is compared with the macroblock whose motion vector is being sought. As in the block matching process performed in order to determine the motion vector with full pixel resolution, the macroblock is compared with each of the eight search blocks according to some criterion, for example a SAD. As a result of the comparisons, a minimum SAD value will generally be obtained. Depending on the nature of the motion in the video sequence, this minimum value may correspond to the location specified by the original motion vector (having full-pixel resolution), or it may correspond to a location having a half-pixel resolution. Thus, it is possible to determine whether a motion vector should point to a full-pixel or sub-pixel location and if sub-pixel resolution is appropriate, to determine the correct sub-pixel resolution motion vector. It should also be appreciated that the scheme just described can be extended to other sub-pixel resolutions (for example, one-quarter-pixel resolution) in an entirely analogous fashion.

In practice, the estimation of a sub-pixel value in the reference frame is performed by interpolating the value of the sub-pixel from surrounding pixel values. In general, interpolation of a sub-pixel value $F(x,y)$ situated at a non-integer location $(x, y)=(n+\Delta x, m+\Delta y)$, can be formulated as a two-dimensional operation, represented mathematically as:

$$F(x, y) = \sum_{k=-K}^{K-1} \sum_{l=-L}^{L-1} f(k+K, l+L) F(n+k, m+l) \quad (7)$$

where $f(k,l)$ are filter coefficients and n and m are obtained by truncating x and y , respectively, to integer values. Typically, the filter coefficients are dependent on the x and y values and the interpolation filters are usually so-called 'separable filters', in which case sub-pixel value $F(x,y)$ can be calculated as follows:

$$F(x, y) = \sum_{k=-K}^{K-1} f(k+K) \sum_{l=-K}^{K-1} f(l+K) F(n+k, m+l) \quad (8)$$

The motion vectors are calculated in the encoder. Once the corresponding motion coefficients are transmitted to the decoder, it is a straightforward matter to interpolate the required sub-pixels using an interpolation method identical to that used in the encoder. In this way, a frame following a reference frame in the Frame Memory 24, can be reconstructed from the reference frame and the motion vectors.

The simplest way of applying sub-pixel value interpolation in a video coder is to interpolate each sub-pixel value every time it is needed. However, this is not an efficient solution in a video encoder, because it is likely that the same sub-pixel value will be required several times and thus calculations to interpolate the same sub-pixel value will be performed multiple times. This results in an unnecessary increase of computational complexity/burden in the encoder.

An alternative approach, which limits the complexity of the encoder, is to pre-calculate and store all sub-pixel values in a memory associated with the encoder. This solution is called interpolation 'before-hand' interpolation hereafter in this document. While limiting complexity, before-hand

interpolation has the disadvantage of increasing memory usage by a large margin. For example, if the motion vector accuracy is one quarter pixel in both horizontal and vertical dimensions, storing pre-calculated sub-pixel values for a complete image results in a memory usage that is 16 times that required to store the original, non-interpolated image. In addition, it involves the calculation of some sub-pixels which might not actually be required in calculating motion vectors in the encoder. Before-hand interpolation is also particularly inefficient in a video decoder, as the majority of pre-calculated sub-pixel values will never be required by the decoder. Thus, it is advantageous not to use pre-calculation in the decoder.

So-called 'on-demand' interpolation can be used to reduce memory requirements in the encoder. For example, if the desired pixel precision is quarter pixel resolution, only sub-pixels at one half unit resolution are interpolated before-hand for the whole frame and stored in the memory. Values of one-quarter pixel resolution sub-pixels are only calculated during the motion estimation/compensation process as and when it is required. In this case memory usage is only 4 times that required to store the original, non-interpolated image.

It should be noted that when before-hand interpolation is used, the interpolation process constitutes only a small fraction of the total encoder computational complexity/burden, since every pixel is interpolated just once. Therefore, in the encoder, the complexity of the interpolation process itself is not very critical when before-hand sub-pixel value interpolation is used. On the other hand, on-demand interpolation poses a much higher computational burden on the encoder, since sub-pixels may be interpolated many times. Hence the complexity of interpolation process, which may be considered in terms of the number of computational operations or operational cycles that must be performed in order to interpolate the sub-pixel values, becomes an important consideration.

In the decoder, the same sub-pixel values are used a few times at most and some are not needed at all. Therefore, in the decoder it is advantageous not to use before-hand interpolation at all, that is, it is advantageous not to pre-calculate any sub-pixel values.

Two interpolation schemes have been developed as part of the work ongoing in the ITU-Telecommunications Standardization Sector, Study Group 16, Video Coding Experts Group (VCEG), Questions 6 and 15. These approaches were proposed for incorporation into ITU-T recommendation H.26L and have been implemented in test models (TML) for the purposes of evaluation and further development. The test model corresponding to Question 15 is referred to as Test Model 5 (TML5), while that resulting from Question 6 is known as Test Model 6 (TML6). The interpolation schemes proposed in both TML5 and TML6 will now be described.

Throughout the description of the sub-pixel value interpolation scheme used in test model TML5, reference will be made to FIG. 12a, which defines a notation for describing pixel and sub-pixel locations specific to TML5. A separate notation, defined in FIG. 13a, will be used in the discussion of the sub-pixel value interpolation scheme used in TML6. A still further notation, illustrated in FIG. 14a, will be used later in the text in connection with the sub-pixel value interpolation method according to the invention. It should be appreciated that the three different notations used in the text are intended to assist in the understanding of each interpolation method and to help distinguish differences between them. However, in all three figures, the letter A is used to denote original image pixels (full pixel resolution). More

specifically, the letter A represents the location of pixels in the image data representing a frame of a video sequence, the pixel values of pixels A being either received as current frame $I_n(x,y)$ from a video source, or reconstructed and stored as a reference frame $R_n(x,y)$ in the Frame Memory 17, 24 of the encoder 10 or the decoder 20. All other letters represent sub-pixel locations, the values of the sub-pixels situated at the sub-pixel locations being obtained by interpolation.

Certain other terms will also be used in a consistent manner throughout the text to identify particular pixel and sub-pixel locations. These are as follows:

The term 'unit horizontal location' is used to describe the location of any sub-pixel that is constructed in a column of the original image data. Sub-pixels c and e in FIGS. 12a and 13a, as well as sub-pixels b and e in FIG. 14a have unit horizontal locations.

The term 'unit vertical location' is used to describe any sub-pixel that is constructed in a row of the original image data. Sub-pixels b and d in FIGS. 12a and 13a as well as sub-pixels b and d in FIG. 14a have unit vertical locations.

By definition, pixels A have unit horizontal and unit vertical locations.

The term 'half horizontal location' is used to describe the location of any sub-pixel that is constructed in a column that lies at half pixel resolution. Sub-pixels b, c, and e shown in FIGS. 12a and 13a fall into this category, as do sub-pixels b, c and f in FIG. 14a. In a similar manner, the term 'half vertical location' is used to describe the location of any sub-pixel that is constructed in a row that lies at half-pixel resolution, such as sub-pixels c and d in FIGS. 12a and 13a, as well as sub-pixels b, c and g in FIG. 14a.

Furthermore, the term 'quarter horizontal location' refers to any sub-pixel that is constructed in a column which lies at quarter-pixel resolution, such as sub-pixels d and e in FIG. 12a, sub-pixels d and g in FIG. 13a and sub-pixels d, g and h in FIG. 14a. Analogously, the term 'quarter vertical location' refers to sub-pixels that are constructed in a row which lies at quarter-pixel resolution. In FIG. 12a, sub-pixels e and f fall into this category, as do sub-pixels e, f and g in FIG. 13a and sub-pixels e, f and h in FIG. 14a.

The definition of each of the terms described above is shown by 'envelopes' drawn on the corresponding figures.

It should further be noted that it is often convenient to denote a particular pixel with a two-dimensional reference. In this case, the appropriate two-dimensional reference can be obtained by examining the intersection of the envelopes in FIGS. 12a, 13a and 14a. Applying this principle, pixel d in FIG. 12a, for example, has a half horizontal and half vertical location and sub-pixel e has a unit horizontal and quarter vertical location. In addition, and for ease of reference, sub-pixels that reside at half unit horizontal and unit vertical locations, unit horizontal and half unit vertical locations as well as half unit horizontal and half unit vertical locations, will be referred to as $\frac{1}{2}$ resolution sub-pixels. Sub-pixels which reside at any quarter unit horizontal and/or quarter unit vertical location will be referred to as $\frac{1}{4}$ resolution sub-pixels.

It should also be noted that in the descriptions of the two test models and in the detailed description of the invention itself, it will be assumed that pixels have a minimum value of 0 and a maximum value of 2^n-1 where n is the number of bits reserved for a pixel value. The number of bits is typically 8. After a sub-pixel has been interpolated, if the value of that interpolated sub-pixel exceeds the value of 2^n-1 , it is restricted to the range of $[0, 2^n-1]$, i.e. values lower than the minimum allowed value will become the

minimum value (0) and values larger than the maximum will become maximum value (2^n-1). This operation is called clipping.

The sub-pixel value interpolation scheme according to TML5 will now be described in detail with reference to FIGS. 12a, 12b and 12c.

1. The value for the sub-pixel at half unit horizontal and unit vertical location, that is $\frac{1}{2}$ resolution sub-pixel b in FIG. 12a, is calculated using a 6-tap filter. The filter interpolates a value for $\frac{1}{2}$ resolution sub-pixel b based upon the values of the 6 pixels (A_1 to A_6) situated in a row at unit horizontal locations and unit vertical locations symmetrically about b, as shown in FIG. 12b, according to the formula $b=(A_1-5A_2+20A_3+20A_4-5A_5+A_6+16)/32$. The operator / denotes division with truncation. The result is clipped to lie in the range $[0, 2^n-1]$.

2. Values for the $\frac{1}{2}$ resolution sub-pixels labelled c are calculated using the same six tap filter as used in step 1 and the six nearest pixels or sub-pixels (A or b) in the vertical direction. Referring now to FIG. 12c, the filter interpolates a value for the $\frac{1}{2}$ resolution sub-pixel c located at unit horizontal and half vertical location based upon the values of the 6 pixels (A_1 to A_6) situated in a column at unit horizontal locations and unit vertical locations symmetrically about c, according to the formula $c=(A_1-5A_2+20A_3+20A_4-5A_5+A_6+16)/32$. Similarly, a value for the $\frac{1}{2}$ resolution sub-pixel c at half horizontal and half vertical location is calculated according to $c=(b_1-5b_2+20b_3+20b_4-5b_5+b_6+16)/32$. Again, the operator / denotes division with truncation. The values calculated for the c sub-pixels are further clipped to lie in the range $[0, 2^n-1]$.

At this point in the interpolation process the values of all $\frac{1}{2}$ resolution sub-pixels have been calculated and the process proceeds to the calculation of $\frac{1}{4}$ resolution sub-pixel values.

3. Values for the $\frac{1}{4}$ resolution sub-pixels labelled d are calculated using linear interpolation and the values of the nearest pixels and/or $\frac{1}{2}$ resolution sub-pixels in the horizontal direction. More specifically, values for $\frac{1}{4}$ resolution sub-pixels d located at quarter horizontal and unit vertical locations, are calculated by taking the average of the immediately neighbouring pixel at unit horizontal and unit vertical location (pixel A) and the immediately neighbouring $\frac{1}{2}$ resolution sub-pixel at half horizontal and unit vertical location (sub-pixel b), i.e. according to $d=(A+b)/2$. Values for $\frac{1}{4}$ resolution sub-pixels d located at quarter horizontal and half vertical locations, are calculated by taking the average of the immediately neighbouring $\frac{1}{2}$ resolution sub-pixels c which lie at unit horizontal and half vertical location and half horizontal and half vertical locations respectively, i.e. according to $d=(c_1+c_2)/2$. Again operator / indicates division with truncation.

4. Values for the $\frac{1}{4}$ resolution sub-pixels labelled e are calculated using linear interpolation and the values of the nearest pixels and/or $\frac{1}{2}$ resolution sub-pixels in the vertical direction. In particular, $\frac{1}{4}$ resolution sub-pixels e at unit horizontal and quarter vertical locations are calculated by taking the average of the immediately neighbouring pixel at unit horizontal and unit vertical location (pixel A) and the immediately neighbouring sub-pixel at unit horizontal and half vertical location (sub-pixel c) according to $e=(A+c)/2$. $\frac{1}{4}$ resolution sub-pixels e_3 at half horizontal and quarter vertical locations are calculated by taking the average of the immediately neighbouring sub-pixel at half horizontal and unit vertical location (sub-pixel b) and the immediately neighbouring sub-pixel at half horizontal and half vertical location (sub-pixel c),

11

according to $e=(b+c)/2$. Furthermore, $\frac{1}{4}$ resolution sub-pixels e at quarter horizontal and quarter vertical locations are calculated by taking the average of the immediately neighbouring sub-pixels at quarter horizontal and unit vertical location and the corresponding sub-pixel at quarter horizontal and half vertical location (sub-pixels d), according to $e=(d_1+d_2)/2$. Once more, operator $/$ indicates division with truncation.

5. The value for $\frac{1}{4}$ resolution sub-pixel f is interpolated by averaging the values of the 4 closest pixels values at unit horizontal and vertical locations, according to $f=(A_1+A_2+A_3+A_4+2)/4$, where pixels A_1 , A_2 , A_3 and A_4 are the four nearest original pixels.

A disadvantage of TML5 is that the decoder is computationally complex. This results from the fact that TML5 uses an approach in which interpolation of $\frac{1}{4}$ resolution sub-pixel values depends upon the interpolation of $\frac{1}{2}$ resolution sub-pixel values. This means that in order to interpolate the values of the $\frac{1}{4}$ resolution sub-pixels, the values of the $\frac{1}{2}$ resolution sub-pixels from which they are determined must be calculated first. Furthermore, since the values of some of the $\frac{1}{4}$ resolution sub-pixels depend upon the interpolated values obtained for other $\frac{1}{4}$ resolution sub-pixels, truncation of the $\frac{1}{4}$ resolution sub-pixel values has a deleterious effect on the precision of some of the $\frac{1}{4}$ resolution sub-pixel values. Specifically, the $\frac{1}{4}$ resolution sub-pixel values are less precise than they would be if calculated from values that had not been truncated and clipped. Another disadvantage of TML5 is that it is necessary to store the values of the $\frac{1}{2}$ resolution sub-pixels in order to interpolate the $\frac{1}{4}$ resolution sub-pixel values. Therefore, excess memory is required to store a result which is not ultimately required.

The sub-pixel value interpolation scheme according to TML6, referred to herein as direct interpolation, will now be described. In the encoder the interpolation method according to TML6 works like the previously described TML5 interpolation method, except that maximum precision is retained throughout. This is achieved by using intermediate values which are neither rounded nor clipped. A step-by-step description of interpolation method according to TML6 as applied in the encoder is given below with reference to FIGS. 13a, 13b and 13c.

1. The value for the sub-pixel at half unit horizontal and unit vertical location, that is $\frac{1}{2}$ resolution sub-pixel b in FIG. 13a, is obtained by first calculating an intermediate value b using a six tap filter. The filter calculates b based upon the values of the 6 pixels (A_1 to A_6) situated in a row at unit horizontal locations and unit vertical locations symmetrically about b , as shown in FIG. 13b, according to the formula $b=(A_1-5A_2+20A_3+20A_4-5A_5+A_6)$. The final value of b is then calculated as $b=(b+16)/32$ and is clipped to lie in the range $[0, 2^n-1]$. As before, the operator $/$ denotes division with truncation.
2. Values for the $\frac{1}{2}$ resolution sub-pixels labelled c are obtained by first calculating intermediate values c . Referring to FIG. 13c, an intermediate value c for the $\frac{1}{2}$ resolution sub-pixel c located at unit horizontal and half vertical location is calculated based upon the values of the 6 pixels (A_1 to A_6) situated in a column at unit horizontal locations and unit vertical locations symmetrically about c , according to the formula $c=(A_1-5A_2+20A_3+20A_4-5A_5+A_6)$. The final value for the $\frac{1}{2}$ resolution sub-pixel c located at unit horizontal and half vertical location is then calculated according to $c=(c+16)/32$. Similarly, an intermediate value c for the $\frac{1}{2}$ resolution sub-pixel c at half horizontal and half vertical location is calculated according to $c=(b_1-5b_2+20b_3+20b_4-5b_5+b_6)$. A final value for

12

this $\frac{1}{2}$ resolution sub-pixel is then calculated according to $(c+512)/1024$. Again, the operator $/$ denotes division with truncation and the values calculated for $\frac{1}{2}$ resolution sub-pixels c are further clipped to lie in the range $[0, 2^n-1]$.

3. Values for the $\frac{1}{4}$ resolution sub-pixels labelled d are calculated as follows. Values for $\frac{1}{4}$ resolution sub-pixels d located at quarter horizontal and unit vertical locations, are calculated from the value of the immediately neighbouring pixel at unit horizontal and unit vertical location (pixel A) and the intermediate value b calculated in step (1) for the immediately neighbouring $\frac{1}{2}$ resolution sub-pixel at half horizontal and unit vertical location ($\frac{1}{2}$ resolution sub-pixel b), according to $d=(32A+b+32)/64$. Values for $\frac{1}{4}$ resolution sub-pixels d located at quarter horizontal and half vertical locations, are interpolated using the intermediate values c calculated for the immediately neighbouring $\frac{1}{2}$ resolution sub-pixels c which lie at unit horizontal and half vertical location and half horizontal and half vertical locations respectively, according to $d=(32c_1+c_2+1024)/2048$. Again operator $/$ indicates division with truncation and the finally obtained $\frac{1}{4}$ resolution sub-pixel values d are clipped to lie in the range $[0, 2^n-1]$.
 4. Values for the $\frac{1}{4}$ resolution sub-pixels labelled e are calculated as follows. Values for $\frac{1}{4}$ resolution sub-pixels e located at unit horizontal and quarter vertical locations are calculated from the value of the immediately neighbouring pixel at unit horizontal and unit vertical location (pixel A) and the intermediate value c calculated in step (2) for the immediately neighbouring $\frac{1}{2}$ resolution sub-pixel at unit horizontal and half vertical location, according to $e=(32A+c+32)/64$. Values for $\frac{1}{4}$ resolution sub-pixels e located at half horizontal and quarter vertical locations are calculated from the intermediate value b calculated in step (1) for the immediately neighbouring $\frac{1}{2}$ resolution sub-pixel at half horizontal and unit vertical location and the intermediate value c calculated in step (2) for the immediately neighbouring $\frac{1}{2}$ resolution sub-pixel at half horizontal and half vertical location, according to $e=(32b+c+1024)/2048$. Once more, operator $/$ indicates division with truncation and the finally obtained $\frac{1}{4}$ resolution sub-pixel values e are clipped to lie in the range $[0, 2^n-1]$.
 5. Values for $\frac{1}{4}$ resolution sub-pixels labelled g are computed using the value of the nearest original pixel A and the intermediate values of the three nearest neighbouring $\frac{1}{2}$ resolution sub-pixels, according to $g=(1024A+32b+32c_1+c_2+2048)/4096$. As before, operator $/$ indicates division with truncation and the finally obtained for $\frac{1}{4}$ resolution sub-pixel values g are clipped to lie in the range $[0, 2^n-1]$.
 6. The value for $\frac{1}{4}$ resolution sub-pixel f is interpolated by averaging the values of the 4 closest pixels at unit horizontal and vertical locations, according to $f=(A_1+A_2+A_3+A_4+2)/4$, where the locations of pixels A_1 , A_2 , A_3 and A_4 are the four nearest original pixels.
- In the decoder, sub-pixel values can be obtained directly by applying 6-tap filters in horizontal and vertical directions. In the case of $\frac{1}{4}$ sub-pixel resolution, referring to FIG. 13a, the filter coefficients applied to pixels and sub-pixels at unit vertical location are $[0, 0, 64, 0, 0, 0]$ for a set of six pixels A , $[1, -5, 52, 20, -5, 1]$ for a set of six sub-pixels d , $[2, -10, 40, 40, -10, 2]$ for a set of six sub-pixels b , and $[1, -5, 20, 52, -5, 1]$ for a set of six sub-pixels d . These filter coefficients are applied to respective sets of pixels or sub-pixels in the same row as the sub-pixel values being interpolated.

13

After applying the filters in the horizontal and vertical directions, interpolated value c is normalized according to $c=(c+2048)/4096$ and clipped to lie in the range $[0, 2^n-1]$. When a motion vector points to an integer pixel position in either the horizontal or vertical direction, many zero coefficients are used. In a practical implementation of TML6, different branches are used in the software which are optimised for the different sub-pixel cases so that there are no multiplications by zero coefficients.

It should be noted that in TML6, $\frac{1}{4}$ resolution sub-pixel values are obtained directly using the intermediate values referred to above and are not derived from rounded and clipped values for $\frac{1}{2}$ resolution sub-pixels. Therefore, in obtaining the $\frac{1}{4}$ resolution sub-pixel values, it is not necessary to calculate final values for any of the $\frac{1}{2}$ resolution sub-pixels. Specifically, it is not necessary to carry out the truncation and clipping operations associated with the calculation of final values for the $\frac{1}{2}$ resolution sub-pixels. Neither is it necessary to have stored final values for $\frac{1}{2}$ resolution sub-pixels for use in the calculation of the $\frac{1}{4}$ resolution sub-pixel values. Therefore TML6 is computationally less complex than TML5, as fewer truncating and clipping operations are required. However, a disadvantage of TML6 is that high precision arithmetic is required both in the encoder and in the decoder. High precision interpolation requires more silicon area in ASICs and requires more computations in some CPUs. Furthermore, implementation of direct interpolation as specified in TML6 in an on-demand fashion has a high memory requirement. This is an important factor, particularly in embedded devices.

In view of the previously presented discussion, it should be appreciated that due to the different requirements of the video encoder and decoder with regard to sub-pixel interpolation, there exists a significant problem in developing a method of sub-pixel value interpolation capable of providing satisfactory performance in both the encoder and decoder. Furthermore, neither of the current test models (TML5, TML6) described in the foregoing can provide a solution that is optimum for application in both encoder and decoder.

SUMMARY OF THE INVENTION

According to a first aspect of the invention there is provided method of interpolation in video coding in which an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, is interpolated to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $\frac{1}{2}^x$, where x is a positive integer having a maximum value N , the method comprising:

a) when values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations are required, interpolating such values directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;

b) when values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations are required, interpolating such values directly using a choice of a first weighted sum of values for sub-pixels residing at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a); and

c) when a value for a sub-pixel situated at a $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical location is required, inter-

14

polating such a value by taking a weighted average of the value of a first sub-pixel or pixel situated at a $\frac{1}{2}^{N-m}$ unit horizontal and $\frac{1}{2}^{N-n}$ unit vertical location and the value of a second sub-pixel or pixel located at a $\frac{1}{2}^{N-p}$ unit horizontal and $\frac{1}{2}^{N-q}$ unit vertical location, variables m , n , p and q taking integer values in the range 1 to N such that the first and second sub-pixels or pixels are located diagonally with respect to the sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ vertical location.

Preferably a first and a second weight are used in the weighted average referred to in (c), the relative magnitudes of the weights being inversely proportional to the (straight-line diagonal) proximity of the first and the second sub-pixel or pixel to the sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ vertical location.

In a situation where the first and the second sub-pixel or pixel are symmetrically located with respect to (equidistant from) the sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ vertical location, the first and second weights may have equal values.

The first weighted sum of values for sub-pixels residing at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations in step b) may be used when a sub-pixel at $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^N$ unit vertical location is required.

The second weighted sum of values for sub-pixels residing at unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations in step b) may be used when a sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location is required.

In one embodiment, when values for sub-pixels at $\frac{1}{2}^N$ unit horizontal and unit vertical locations, and $\frac{1}{2}^N$ horizontal and $\frac{1}{2}^{N-1}$ vertical locations are required, such values are interpolated by taking the average of the values of a first pixel or sub-pixel located at a vertical location corresponding to that of the sub-pixel being calculated and unit horizontal location and a second pixel or sub-pixel located at a vertical location corresponding to that of the sub-pixel being calculated and $\frac{1}{2}^{N-1}$ unit horizontal location.

When values for sub-pixels at unit horizontal and $\frac{1}{2}^N$ unit vertical locations, and $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations are required, they may be interpolated by taking the average of the values of a first pixel or sub-pixel located at a horizontal location corresponding to that of the sub-pixel being calculated and unit vertical location and a second pixel or sub-pixel located at a horizontal location corresponding to that of the sub-pixel being calculated and $\frac{1}{2}^{N-1}$ unit vertical location.

Values for sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations may be interpolated by taking the average of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

Values for sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations may be interpolated by taking the average of values of a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

Values for half of the sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations may be interpolated by taking the average of a first pair of values of a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location and values for the other half of the sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations are interpolated by taking the average of a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

15

Values for sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations are alternately interpolated for one such sub-pixel by taking the average of a first pair of values of a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location and values and for a neighbouring such sub-pixel by taking the average of a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

The sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations may be alternately interpolated in a horizontal direction.

The sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations may be alternately interpolated in a horizontal direction.

When values for some sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations are required, such values may be interpolated by taking the average of a plurality of nearest neighbouring pixels.

At least one of step a) and step b) interpolating sub-pixel values directly using weighted sums may involve the calculation of an intermediate value for the sub-pixel values having a dynamic range greater than the specified dynamic range.

The intermediate value for a sub-pixel having $\frac{1}{2}^{N-1}$ sub-pixel resolution may be used the calculation of a sub-pixel value having $\frac{1}{2}^N$ sub-pixel resolution.

According to a second aspect of the invention, there is provided a method of interpolation in video coding in which an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, is interpolated to generate values for sub-pixels at fractional horizontal and vertical locations, the method comprising:

a) when values for sub-pixels at half unit horizontal and unit vertical locations, and unit horizontal and half unit vertical locations are required, interpolating such values directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;

b) when values for sub-pixels at half unit horizontal and half unit vertical locations are required, interpolating such values directly using a weighted sum of values for sub-pixels residing at half unit horizontal and unit vertical locations calculated according to step (a); and

c) when values for sub-pixels at quarter unit horizontal and quarter unit vertical locations are required, interpolating such values by taking the average of at least one pair of a first pair of values of a sub-pixel located at a half unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and half unit vertical location and a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a half unit horizontal and half unit vertical location.

According to a third aspect of the invention, there is provided a method of interpolation in video coding in which an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, is interpolated to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to

16

$\frac{1}{2}^x$ where x is a positive integer having a maximum value N, the method comprising:

a) when values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations are required, interpolating such values directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;

b) when a value for a sub-pixel at a sub-pixel horizontal and sub-pixel vertical location is required, interpolating such a value directly using a choice of a first weighted sum of values for sub-pixels located at a vertical location corresponding to that of the sub-pixel being calculated and a second weighted sum of values for sub-pixels located at a horizontal location corresponding to that of the sub-pixel being calculated.

The sub-pixels used in the first weighted sum may be sub-pixels residing at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations and the first weighted sum may be used to interpolate a value for a sub-pixel at $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^N$ unit vertical location.

The sub-pixels used in the second weighted sum may be sub-pixels residing at unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations and the second weighted sum may be used to interpolate a value for a sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

When values for sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations are required, they may be interpolated by taking the average of at least one pair of a first pair of values of a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location and a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

In the foregoing, N may be equal an integer selected from a list consisting of the values 2, 3, and 4.

Sub-pixels at quarter unit horizontal location are to be interpreted as being sub-pixels having as their left-hand nearest neighbour a pixel at unit horizontal location and as their right-hand nearest neighbour a sub-pixel at half unit horizontal location as well as sub-pixels having as their left-hand nearest neighbour a sub-pixel at half unit horizontal location and as their right-hand nearest neighbour a pixel at unit horizontal location. Correspondingly, sub-pixels at quarter unit vertical location are to be interpreted as being sub-pixels having as their upper nearest neighbour a pixel at unit vertical location and as their lower nearest neighbour a sub-pixel at half unit vertical location as well as sub-pixels having as their upper nearest neighbour a sub-pixel at half unit vertical location and as their lower nearest neighbour a pixel at unit vertical location.

The term dynamic range, refers to the range of values which the sub-pixel values and the weighted sums can take.

Preferably changing the dynamic range, whether by extending it or reducing it, means changing the number of bits which are used to represent the dynamic range.

In an embodiment of the invention, the method is applied to an image that is sub-divided into a number of image blocks. Preferably each image block comprises four corners, each corner being defined by a pixel located at a unit horizontal and unit vertical location. Preferably the method is applied to each image block as the block becomes available for sub-pixel value interpolation. Alternatively, sub-pixel value interpolation according to the method of the invention is performed once all image blocks of an image have become available for sub-pixel value interpolation.

17

Preferably the method is used in video encoding. Preferably the method is used in video decoding.

In one embodiment of the invention, when used in encoding, the method is carried out as before-hand interpolation, in which values for all sub-pixels at half unit locations and values for all sub-pixels at quarter unit locations are calculated and stored before being subsequently used in the determination of a prediction frame during motion predictive coding. In alternative embodiments, the method is carried out as a combination of before-hand and on-demand interpolation. In this case, a certain proportion or category of sub-pixel values is calculated and stored before being used in the determination of a prediction frame and certain other sub-pixel values are calculated only when required during motion predictive coding.

Preferably, when the method is used in decoding, sub-pixels are only interpolated when their need is indicated by a motion vector.

According to a fourth aspect of the invention, there is provided a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the video coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $\frac{1}{2}^x$, where x is a positive integer having a maximum value N, the interpolator being adapted to:

a) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;

b) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using a choice of a first weighted sum of values for sub-pixels residing at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a); and

c) interpolate a value for a sub-pixel situated at a $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical location by taking a weighted average of the value of a first sub-pixel or pixel situated at a $\frac{1}{2}^{N-m}$ unit horizontal and $\frac{1}{2}^{N-n}$ unit vertical location and the value of a second sub-pixel or pixel located at a $\frac{1}{2}^{N-p}$ unit horizontal and $\frac{1}{2}^{N-q}$ unit vertical location, variables m, n, p and q taking integer values in the range 1 to N such that the first and second sub-pixels or pixels are located diagonally with respect to the sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ vertical location.

The video coder may comprise a video encoder. It may comprise a video decoder. There may be a codec comprising both the video encoder and the video decoder.

According to a fifth aspect of the invention, there is provided a communications terminal comprising a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the video coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to

18

$\frac{1}{2}^x$, where x is a positive integer having a maximum value N, the interpolator being adapted to:

a) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;

b) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using a choice of a first weighted sum of values for sub-pixels residing at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a); and

c) interpolate a value for a sub-pixel situated at a $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical location by taking a weighted average of the value of a first sub-pixel or pixel situated at a $\frac{1}{2}^{N-m}$ unit horizontal and $\frac{1}{2}^{N-n}$ unit vertical location and the value of a second sub-pixel or pixel located at a $\frac{1}{2}^{N-p}$ unit horizontal and $\frac{1}{2}^{N-q}$ unit vertical location, variables m, n, p and q taking integer values in the range 1 to N such that the first and second sub-pixels or pixels are located diagonally with respect to the sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ vertical location.

The communications terminal may comprise a video encoder. It may comprise a video decoder. Preferably, it comprises a video codec comprising a video encoder and a video decoder.

Preferably the communications terminal comprising a user interface, a processor and at least one of a transmitting block and a receiving block, and a video coder according to at least one of the third and fourth aspects of the invention. Preferably the processor controls the operation of the transmitting block and/or the receiving block and the video coder.

According to a sixth aspect of the invention, there is provided a telecommunications system comprising a communications terminal and a network, the telecommunications network and the communications terminal being connected by a communications link over which coded video can be transmitted, the communications terminal comprising a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the video coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $\frac{1}{2}^x$, where x is a positive integer having a maximum value N, the interpolator being adapted to:

a) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;

b) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using a choice of a first weighted sum of values for sub-pixels residing at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a); and

c) interpolate a value for a sub-pixel situated at a $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical location by taking a weighted average of the value of a first sub-pixel or pixel situated at a $\frac{1}{2}^{N-m}$ unit horizontal and $\frac{1}{2}^{N-n}$ unit vertical location and

the value of a second sub-pixel or pixel located at a $\frac{1}{2}^{N-p}$ unit horizontal and $\frac{1}{2}^{N-q}$ unit vertical location, variables m, n, p and q taking integer values in the range 1 to N such that the first and second sub-pixels or pixels are located diagonally with respect to the sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ vertical location.

Preferably the telecommunications system is a mobile telecommunications system comprising a mobile communications terminal and a wireless network, the connection between the mobile communications terminal and the wireless network being formed by a radio link. Preferably the network enables the communications terminal to communicate with other communications terminals connected to the network over communications links between the other communications terminals and the network.

According to a seventh aspect of the invention, there is provided a telecommunications system comprising a communications terminal and a network, the telecommunications network and the communications terminal being connected by a communications link over which coded video can be transmitted, the network comprising a video coder for coding for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the video coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $\frac{1}{2}^x$, where x is a positive integer having a maximum value N, the interpolator being adapted to:

a) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;

b) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using a choice of a first weighted sum of values for sub-pixels residing at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a); and

c) interpolate a value for a sub-pixel situated at a $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical location by taking a weighted average of the value of a first sub-pixel or pixel situated at a $\frac{1}{2}^{N-m}$ unit horizontal and $\frac{1}{2}^{N-n}$ unit vertical location and the value of a second sub-pixel or pixel located at a $\frac{1}{2}^{N-p}$ unit horizontal and $\frac{1}{2}^{N-q}$ unit vertical location, variables m, n, p and q taking integer values in the range 1 to N such that the first and second sub-pixels or pixels are located diagonally with respect to the sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ vertical location.

According to an eighth aspect of the invention there is provided a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the resolution of the sub-pixels being determined by a positive integer N, the interpolator being adapted to:

a) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;

b) interpolate a value for a sub-pixel at a sub-pixel horizontal and sub-pixel vertical location is required directly using a choice of a first weighted sum of values for sub-pixels located at a vertical location corresponding to that of the sub-pixel being calculated and a second weighted sum of values for sub-pixels located at a horizontal location corresponding to that of the sub-pixel being calculated.

The interpolator may be further adapted to form the first weighted sum using the values of sub-pixels residing at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations and to use the first weighted sum to interpolate a value for a sub-pixel at $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^N$ unit vertical location.

The interpolator may be further adapted to form the second weighted sum using the values of sub-pixels residing at unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations and to use the second weighted sum to interpolate a value for a sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

The interpolator may be further adapted to interpolate values for sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations by taking the average of at least one pair of a first pair of values of a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location and a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

According to a ninth aspect of the invention there is provided a communications terminal comprising a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the resolution of the sub-pixels being determined by a positive integer N, the interpolator being adapted to:

a) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;

b) interpolate a value for a sub-pixel at a sub-pixel horizontal and sub-pixel vertical location is required directly using a choice of a first weighted sum of values for sub-pixels located at a vertical location corresponding to that of the sub-pixel being calculated and a second weighted sum of values for sub-pixels located at a horizontal location corresponding to that of the sub-pixel being calculated.

According to a tenth aspect of the invention there is provided a telecommunications system comprising a communications terminal and a network, the telecommunications network and the communications terminal being connected by a communications link over which coded video can be transmitted, the communications terminal comprising a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the resolution of the sub-pixels being determined by a positive integer N, the interpolator being adapted to:

a) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;

b) interpolate a value for a sub-pixel at a sub-pixel horizontal and sub-pixel vertical location is required directly using a choice of a first weighted sum of values for sub-pixels located at a vertical location corresponding to that of the sub-pixel being calculated and a second weighted sum of values for sub-pixels located at a horizontal location corresponding to that of the sub-pixel being calculated.

According to an eleventh aspect of the invention there is provided a telecommunications system comprising a communications terminal and a network, the telecommunications network and the communications terminal being connected by a communications link over which coded video can be transmitted, the network comprising a video coder for coding for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the resolution of the sub-pixels being determined by a positive integer N, the interpolator being adapted to:

a) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and $\frac{1}{2}^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;

b) interpolate a value for a sub-pixel at a sub-pixel horizontal and sub-pixel vertical location is required directly using a choice of a first weighted sum of values for sub-pixels located at a vertical location corresponding to that of the sub-pixel being calculated and a second weighted sum of values for sub-pixels located at a horizontal location corresponding to that of the sub-pixel being calculated.

BRIEF DESCRIPTION OF THE FIGURES

An embodiment of the invention will now be described by way of example only with reference to the accompanying drawings in which:

FIG. 1 shows a video encoder according to the prior art;

FIG. 2 shows a video decoder according to the prior art;

FIG. 3 shows the types of frames used in video encoding;

FIGS. 4a, 4b, and 4c show steps in block-matching;

FIG. 5 illustrates the process of motion estimation to sub-pixel resolution;

FIG. 6 shows a terminal device comprising video encoding and decoding equipment in which the method of the invention may be implemented;

FIG. 7 shows a video encoder according an embodiment of the present invention;

FIG. 8 shows a video decoder according to an embodiment of the present invention;

FIG. 11 shows a schematic diagram of a mobile telecommunications network according to an embodiment of the present invention;

FIG. 12a shows a notation for describing pixel and sub-pixel locations specific to TML5;

FIG. 12b shows interpolation of a half resolution sub-pixels;

FIG. 12c shows interpolation of a half resolution sub-pixels;

FIG. 13a shows a notation for describing pixel and sub-pixel locations specific to TML6;

FIG. 13b shows interpolation of a half resolution sub-pixels;

FIG. 13c shows interpolation of a half resolution sub-pixels;

FIG. 14 shows a notation for describing pixel and sub-pixel locations specific to the invention;

FIG. 14b shows interpolation of a half resolution sub-pixels according to the invention;

FIG. 14c shows interpolation of a half resolution sub-pixels according to the invention;

FIG. 15 shows possible choices of diagonal interpolation for sub-pixels;

FIG. 16 shows the half resolution sub-pixel values required to calculate other half resolution sub-pixel values;

FIG. 17a shows the half resolution sub-pixel values that must be calculated in order to interpolate values for quarter resolution sub-pixels in an image block using the interpolation method of TML5;

FIG. 17b shows the half resolution sub-pixel values that must be calculated in order to interpolate values for quarter resolution sub-pixels in an image block using the interpolation method according to the invention;

FIG. 18a shows the numbers of half resolution sub-pixels that must be calculated in order to obtain values for quarter resolution sub-pixels within an image block using the sub-pixel value interpolation method according to TML5;

FIG. 18b shows the numbers of half resolution sub-pixels that must be calculated in order to obtain values for quarter resolution sub-pixels within an image block using the sub-pixel value interpolation method according to the invention;

FIG. 19 shows a numbering scheme for each of the 15 sub-pixel positions;

FIG. 20 shows nomenclature used to describe pixels, half resolution sub-pixels, quarter resolution sub-pixels and eighth resolution sub-pixels

FIG. 21a shows the diagonal direction to be used in the interpolation of each eighth resolution sub-pixel in an embodiment of the invention;

FIG. 21b shows the diagonal direction to be used in the interpolation of each eighth resolution sub-pixel in another embodiment of the invention; and

FIG. 22 shows nomenclature used to describe eighth resolution sub-pixels within an image.

DETAILED DESCRIPTION

FIGS. 1 to 5, 12a, 12b, 12c, 13a, 13b, and 13c have been described in the foregoing.

FIG. 6 presents a terminal device comprising video encoding and decoding equipment which may be adapted to operate in accordance with the present invention. More precisely, the figure illustrates a multimedia terminal 60 implemented according to ITU-T recommendation H.324. The terminal can be regarded as a multimedia transceiver device. It includes elements that capture, encode and multiplex multimedia data streams for transmission via a communications network, as well as elements that receive, de-multiplex, decode and display received multimedia content. ITU-T recommendation H.324 defines the overall operation of the terminal and refers to other recommendations that govern the operation of its various constituent parts. This kind of multimedia terminal can be used in real-time applications such as conversational videotelephony, or non real-time applications such as the retrieval/streaming of video clips, for example from a multimedia content server in the Internet.

In the context of the present invention, it should be appreciated that the H.324 terminal shown in FIG. 6 is only

one of a number of alternative multimedia terminal implementations suited to application of the inventive method. It should also be noted that a number of alternatives exist relating to the location and implementation of the terminal equipment. As illustrated in FIG. 6, the multimedia terminal may be located in communications equipment connected to a fixed line telephone network such as an analogue PSTN (Public Switched Telephone Network). In this case the multimedia terminal is equipped with a modem 71, compliant with ITU-T recommendations V.8, V.34 and optionally V.8bis. Alternatively, the multimedia terminal may be connected to an external modem. The modem enables conversion of the multiplexed digital data and control signals produced by the multimedia terminal into an analogue form suitable for transmission over the PSTN. It further enables the multimedia terminal to receive data and control signals in analogue form from the PSTN and to convert them into a digital data stream that can be demultiplexed and processed in an appropriate manner by the terminal.

An H.324 multimedia terminal may also be implemented in such a way that it can be connected directly to a digital fixed line network, such as an ISDN (Integrated Services Digital Network). In this case the modem 71 is replaced with an ISDN user-network interface. In FIG. 6, this ISDN user-network interface is represented by alternative block 72.

H.324 multimedia terminals may also be adapted for use in mobile communication applications. If used with a wireless communication link, the modem 71 can be replaced with any appropriate wireless interface, as represented by alternative block 73 in FIG. 6. For example, an H.324/M multimedia terminal can include a radio transceiver enabling connection to the current 2nd generation GSM mobile telephone network, or the proposed 3rd generation UMTS (Universal Mobile Telephone System).

It should be noted that in multimedia terminals designed for two-way communication, that is for transmission and reception of video data, it is advantageous to provide both a video encoder and video decoder implemented according to the present invention. Such an encoder and decoder pair is often implemented as a single combined functional unit, referred to as a 'codec'.

Because a video encoder according to the invention performs motion compensated video encoding to sub-pixel resolution using a specific interpolation scheme and a particular combination of before-hand and on-demand sub-pixel value interpolation, it is generally necessary for a video decoder of a receiving terminal to be implemented in a manner compatible with the encoder of the transmitting terminal which formed the compressed video data stream. Failure to ensure this compatibility may have an adverse effect on the quality of the motion compensation and the accuracy of reconstructed video frames.

A typical H.324 multimedia terminal will now be described in further detail with reference to FIG. 6.

The multimedia terminal 60 includes a variety of elements referred to as 'terminal equipment'. This includes video, audio and telematic devices, denoted generically by reference numbers 61, 62 and 63, respectively. The video equipment 61 may include, for example, a video camera for capturing video images, a monitor for displaying received video content and optional video processing equipment. The audio equipment 62 typically includes a microphone, for example for capturing spoken messages, and a loudspeaker for reproducing received audio content. The audio equipment may also include additional audio processing units.

The telematic equipment 63, may include a data terminal, keyboard, electronic whiteboard or a still image transceiver, such as a fax unit.

The video equipment 61 is coupled to a video codec 65. The video codec 65 comprises a video encoder and a corresponding video decoder both implemented according to the invention. Such an encoder and a decoder will be described in the following. The video codec 65 is responsible for encoding captured video data in an appropriate form for further transmission over a communications link and decoding compressed video content received from the communications network. In the example illustrated in FIG. 6, the video codec is implemented according to ITU-T recommendation H.263, with appropriate modifications to implement the sub-pixel value interpolation method according to the invention in both the encoder and the decoder of the video codec.

Similarly, the terminal's audio equipment is coupled to an audio codec, denoted in FIG. 6 by reference number 66. Like the video codec, the audio codec comprises an encoder/decoder pair. It converts audio data captured by the terminal's audio equipment into a form suitable for transmission over the communications link and transforms encoded audio data received from the network back into a form suitable for reproduction, for example on the terminal's loudspeaker. The output of the audio codec is passed to a delay block 67. This compensates for the delays introduced by the video coding process and thus ensures synchronisation of audio and video content.

The system control block 64 of the multimedia terminal controls end-to-network signalling using an appropriate control protocol (signalling block 68) to establish a common mode of operation between a transmitting and a receiving terminal. The signalling block 68 exchanges information about the encoding and decoding capabilities of the transmitting and receiving terminals and can be used to enable the various coding modes of the video encoder. The system control block 64 also controls the use of data encryption. Information regarding the type of encryption to be used in data transmission is passed from encryption block 69 to the multiplexer/de-multiplexer (MUX/DMUX unit) 70.

During data transmission from the multimedia terminal, the MUX/DMUX unit 70 combines encoded and synchronised video and audio streams with data input from the telematic equipment 63 and possible control data, to form a single bit-stream. Information concerning the type of data encryption (if any) to be applied to the bit-stream, provided by encryption block 69, is used to select an encryption mode. Correspondingly, when a multiplexed and possibly encrypted multimedia bit-stream is being received, MUX/DMUX unit 70 is responsible for decrypting the bit-stream, dividing it into its constituent multimedia components and passing those components to the appropriate codec(s) and/or terminal equipment for decoding and reproduction.

It should be noted that the functional elements of the multimedia terminal, video encoder, decoder and video codec according to the invention can be implemented as software or dedicated hardware, or a combination of the two. The video encoding and decoding methods according to the invention are particularly suited for implementation in the form of a computer program comprising machine-readable instructions for performing the functional steps of the invention. As such, the encoder and decoder according to the invention may be implemented as software code stored on a storage medium and executed in a computer, such as a personal desktop computer, in order to provide that computer with video encoding and/or decoding functionality.

If the multimedia terminal **60** is a mobile terminal, that is if it is equipped with a radio transceiver **73**, it will be understood by those skilled in the art that it may also comprise additional elements. In one embodiment it comprises a user interface having a display and a keyboard, which enables operation of the multimedia terminal **60** by a user, together with necessary functional blocks including a central processing unit, such as a microprocessor, which controls the blocks responsible for different functions of the multimedia terminal, a random access memory RAM, a read only memory ROM, and a digital camera. The microprocessor's operating instructions, that is program code corresponding to the basic functions of the multimedia terminal **60**, is stored in the read-only memory ROM and can be executed as required by the microprocessor, for example under control of the user. In accordance with the program code, the microprocessor uses the radio transceiver **73** to form a connection with a mobile communication network, enabling the multimedia terminal **60** to transmit information to and receive information from the mobile communication network over a radio path.

The microprocessor monitors the state of the user interface and controls the digital camera. In response to a user command, the microprocessor instructs the camera to record digital images into the RAM. Once an image is captured, or alternatively during the capturing process, the microprocessor segments the image into image segments (for example macroblocks) and uses the encoder to perform motion compensated encoding for the segments in order to generate a compressed image sequence as explained in the foregoing description. A user may command the multimedia terminal **60** to display the captured images on its display or to send the compressed image sequence using the radio transceiver **73** to another multimedia terminal, a video telephone connected to a fixed line network (PSTN) or some other telecommunications device. In a preferred embodiment, transmission of image data is started as soon as the first segment is encoded so that the recipient can start a corresponding decoding process with a minimum delay.

FIG. **11** is a schematic diagram of a mobile telecommunications network according to an embodiment of the invention. Multimedia terminals MS are in communication with base stations BTS by means of a radio link. The base stations BTS are further connected, through a so-called Abis interface, to a base station controller BSC, which controls and manages several base stations.

The entity formed by a number of base stations BTS (typically, by a few tens of base stations) and a single base station controller BSC, controlling the base stations, is called a base station subsystem BSS. Particularly, the base station controller BSC manages radio communication channels and handovers. The base station controller BSC is also connected, through a so-called A interface, to a mobile services switching centre MSC, which co-ordinates the formation of connections to and from mobile stations. A further connection is made, through the mobile service switching centre MSC, to outside the mobile communications network. Outside the mobile communications network there may further reside other network(s) connected to the mobile communications network by gateway(s) GTW, for example the Internet or a Public Switched Telephone Network (PSTN). In such an external network, or within the telecommunications network, there may be located video decoding or encoding stations, such as computers PC. In an embodiment of the invention, the mobile telecommunications network comprises a video server VSRVR to provide video data to a MS subscribing to such a service. The video

data is compressed using the motion compensated video compression method as described in the foregoing. The video server may function as a gateway to an online video source or it may comprise previously recorded video clips. Typical videotelephony applications may involve, for example, two mobile stations or one mobile station MS and a videotelephone connected to the PSTN, a PC connected to the Internet or a H.261 compatible terminal connected either to the Internet or to the PSTN.

FIG. **7** shows a video encoder **700** according to an embodiment the invention. FIG. **8** shows a video decoder **800** according to an embodiment the invention.

The encoder **700** comprises an input **701** for receiving a video signal from a camera or other video source (not shown). It further comprises a DCT transformer **705**, a quantiser **706**, an inverse quantiser **709**, an inverse DCT transformer **710**, combiners **712** and **716**, a before-hand sub-pixel interpolation block **730**, a frame store **740** and an on-demand sub-pixel interpolation block **750**, implemented in combination with motion estimation block **760**. The encoder also comprises a motion field coding block **770** and a motion compensated prediction block **780**. Switches **702** and **714** are operated co-operatively by a control manager **720** to switch the encoder between an INTRA-mode of video encoding and an INTER-mode of video encoding. The encoder **700** also comprises a multiplexer unit (MUX/DMUX) **790** to form a single bit-stream from the various types of information produced by the encoder **700** for further transmission to a remote receiving terminal, or for example for storage on a mass storage medium such as a computer hard drive (not shown).

It should be noted that the presence and implementations of before-hand sub-pixel interpolation block **730** and on-demand sub-pixel value interpolation block **750** in the encoder architecture depend on the way in which the sub-pixel interpolation method according to the invention is applied. In embodiments of the invention in which before-hand sub-pixel value interpolation is not performed, encoder **700** does not comprise before-hand sub-pixel value interpolation block **730**. In other embodiments of the invention, only before-hand sub-pixel interpolation is performed and thus the encoder does not include on-demand sub-pixel value interpolation block **750**. In embodiments in which both before-hand and on-demand sub-pixel value interpolation are performed, both blocks **730** and **750** are present in the encoder **700**.

Operation of the encoder **700** according to the invention will now be described in detail. In the description, it will be assumed that each frame of uncompressed video, received from the video source at the input **701**, is received and processed on a macroblock-by-macroblock basis, preferably in raster-scan order. It will further be assumed that when the encoding of a new video sequence starts, the first frame of the sequence is encoded in INTRA-mode. Subsequently, the encoder is programmed to code each frame in INTER-format, unless one of the following conditions is met: 1) it is judged that the current frame being coded is so dissimilar from the reference frame used in its prediction that excessive prediction error information is produced; 2) a predefined INTRA frame repetition interval has expired; or 3) feedback is received from a receiving terminal indicating a request for a frame to be coded in INTRA format.

The occurrence of condition 1) is detected by monitoring the output of the combiner **716**. The combiner **716** forms a difference between the current macroblock of the frame being coded and its prediction, produced in the motion

compensated prediction block **780**. If a measure of this difference (for example a sum of absolute differences of pixel values) exceeds a predetermined threshold, the combiner **716** informs the control manager **720** via a control line **717** and the control manager **720** operates the switches **702** and **714** so as to switch the encoder **700** into INTRA coding mode. Occurrence of condition 2) is monitored by means of a timer or frame counter implemented in the control manager **720**, in such a way that if the timer expires, or the frame counter reaches a predetermined number of frames, the control manager **720** operates the switches **702** and **714** to switch the encoder into INTRA coding mode. Condition 3) is triggered if the control manager **720** receives a feedback signal from, for example, a receiving terminal, via control line **718** indicating that an INTRA frame refresh is required by the receiving terminal. Such a condition might arise, for example, if a previously transmitted frame were badly corrupted by interference during its transmission, rendering it impossible to decode at the receiver. In this situation, the receiver would issue a request for the next frame to be encoded in INTRA format, thus re-initialising the coding sequence.

It will further be assumed that the encoder and decoder are implemented in such a way as to allow the determination of motion vectors with a spatial resolution of up to quarter-pixel resolution. As will be seen in the following, finer levels of resolution are also possible.

Operation of the encoder **700** in INTRA coding mode will now be described. In INTRA-mode, the control manager **720** operates the switch **702** to accept video input from input line **719**. The video signal input is received macroblock by macroblock from input **701** via the input line **719** and each macroblock of original image pixels is transformed into DCT coefficients by the DCT transformer **705**. The DCT coefficients are then passed to the quantiser **706**, where they are quantised using a quantisation parameter QP. Selection of the quantisation parameter QP is controlled by the control manager **720** via control line **722**. Each DCT transformed and quantised macroblock that makes up the INTRA coded image information **723** of the frame is passed from the quantiser **706** to the MUX/DMUX **790**. The MUX/DMUX **790** combines the INTRA coded image information with possible control information (for example header data, quantisation parameter information, error correction data etc.) to form a single bit-stream of coded image information **725**. Variable length coding (VLC) is used to reduce redundancy of the compressed video bit-stream, as is known to those skilled in the art.

A locally decoded picture is formed in the encoder **700** by passing the data output by the quantiser **706** through inverse quantiser **709** and applying an inverse DCT transform **710** to the inverse-quantised data. The resulting data is then input to the combiner **712**. In INTRA mode, switch **714** is set so that the input to the combiner **712** via the switch **714** is set to zero. In this way, the operation performed by the combiner **712** is equivalent to passing the decoded image data formed by the inverse quantiser **709** and the inverse DCT transform **710** unaltered.

In embodiments of the invention in which before-hand sub-pixel value interpolation is performed, the output from combiner **712** is applied to before-hand sub-pixel interpolation block **730**. The input to the before-hand sub-pixel value interpolation block **730** takes the form of decoded image blocks. In the before-hand sub-pixel value interpolation block **730**, each decoded macroblock is subjected to sub-pixel interpolation in such a way that a predetermined sub-set of sub-pixel resolution sub-pixel values is calculated

according to the interpolation method of the invention and is stored together with the decoded pixel values in frame store **740**.

In embodiments in which before-hand sub-pixel interpolation is not performed, before-hand sub-pixel interpolation block is not present in the encoder architecture and the output from combiner **712**, comprising decoded image blocks, is applied directly to frame store **740**.

As subsequent macroblocks of the current frame are received and undergo the previously described coding and decoding steps in blocks **705**, **706**, **709**, **710**, **712**, a decoded version of the INTRA frame is built up in the frame store **740**. When the last macroblock of the current frame has been INTRA coded and subsequently decoded, the frame store **740** contains a completely decoded frame, available for use as a prediction reference frame in coding a subsequently received video frame in INTER format. In embodiments of the invention in which before-hand sub-pixel value interpolation is performed, the reference frame held in frame store **740** is at least partially interpolated to sub-pixel resolution.

Operation of the encoder **700** in INTER coding mode will now be described. In INTER coding mode, the control manager **720** operates switch **702** to receive its input from line **721**, which comprises the output of the combiner **716**. The combiner **716** forms prediction error information representing the difference between the current macroblock of the frame being coded and its prediction, produced in the motion compensated prediction block **780**. The prediction error information is DCT transformed in block **705** and quantised in block **706** to form a macroblock of DCT transformed and quantised prediction error information. Each macroblock of DCT transformed and quantised prediction error information is passed from the quantiser **706** to the MUX/DMUX unit **790**. The MUX/DMUX unit **790** combines the prediction error information **723** with motion coefficients **724** (described in the following) and control information (for example header data, quantisation parameter information, error correction data etc.) to form a single bit-stream of coded image information, **725**.

Locally decoded prediction error information for the each macroblock of the INTER coded frame is then formed in the encoder **700** by passing the encoded prediction error information **723** output by the quantiser **706** through the inverse quantiser **709** and applying an inverse DCT transform in block **710**. The resulting locally decoded macroblock of prediction error information is then input to combiner **712**. In INTER-mode, switch **714** is set so that the combiner **712** also receives motion predicted macroblocks for the current INTER frame, produced in the motion compensated prediction block **780**. The combiner **712** combines these two pieces of information to produce reconstructed image blocks for the current INTER frame.

As described above when considering INTRA coded frames, in embodiments of the invention in which before-hand sub-pixel value interpolation is performed, the output from combiner **712** is applied to the before-hand sub-pixel interpolation block **730**. Thus, the input to the before-hand sub-pixel value interpolation block **730** in INTER coding mode also takes the form of decoded image blocks. In the before-hand sub-pixel value interpolation block **730**, each decoded macroblock is subjected to sub-pixel interpolation in such a way that a predetermined sub-set of sub-pixel values is calculated according to the interpolation method of the invention and is stored together with the decoded pixel values in frame store **740**. In embodiments in which before-hand sub-pixel interpolation is not performed, before-hand

sub-pixel interpolation block is not present in the encoder architecture and the output from combiner 712, comprising decoded image blocks, is applied directly to frame store 740.

As subsequent macroblocks of the video signal are received from the video source and undergo the previously described coding and decoding steps in blocks 705, 706, 709, 710, 712, a decoded version of the INTER frame is built up in the frame store 740. When the last macroblock of the frame has been INTER coded and subsequently decoded, the frame store 740 contains a completely decoded frame, available for use as a prediction reference frame in encoding a subsequently received video frame in INTER format. In embodiments of the invention in which before-hand sub-pixel value interpolation is performed, the reference frame held in frame store 740 is at least partially interpolated to sub-pixel resolution.

Formation of a prediction for a macroblock of the current frame will now be described.

Any frame encoded in INTER format requires a reference frame for motion compensated prediction. This means, inter alia, that when encoding a video sequence, the first frame to be encoded, whether it is the first frame in the sequence, or some other frame, must be encoded in INTRA format. This, in turn, means that when the video encoder 700 is switched into INTER coding mode by control manager 720, a complete reference frame, formed by locally decoding a previously encoded frame, is already available in the frame store 740 of the encoder. In general, the reference frame is formed by locally decoding either an INTRA coded frame or an INTER coded frame.

The first step in forming a prediction for a macroblock of the current frame is performed by motion estimation block 760. The motion estimation block 760 receives the current macroblock of the frame being coded via line 727 and performs a block matching operation in order to identify a region in the reference frame which corresponds substantially with the current macroblock. According to the invention, the block-matching process is performed to sub-pixel resolution in a manner that depends on the implementation of the encoder 700 and the degree of before-hand sub-pixel interpolation performed. However, the basic principle behind the block-matching process is similar in all cases. Specifically, motion estimation block 760 performs block-matching by calculating difference values (e.g. sums of absolute differences) representing the difference in pixel values between the macroblock of the current frame under examination and candidate best-matching regions of pixels/sub-pixels in the reference frame. A difference value is produced for all possible offsets (e.g. quarter- or one eighth sub-pixel precision x, y displacements) between the macroblock of the current frame and candidate test region within a predefined search region of the reference frame and motion estimation block 760 determines the smallest calculated difference value. The offset between the macroblock in the current frame and the candidate test region of pixel values/sub-pixel values in the reference frame that yields the smallest difference value defines the motion vector for the macroblock in question. In certain embodiments of the invention, an initial estimate for the motion vector having unit pixel precision is first determined and then refined to a finer level of sub-pixel precision, as described in the foregoing.

In embodiments of the encoder in which before-hand sub-pixel value interpolation is not performed, all sub-pixel values required in the block matching process are calculated in on-demand sub-pixel value interpolation block 750.

Motion estimation block 760 controls on-demand sub-pixel value interpolation block 750 to calculate each sub-pixel value needed in the block-matching process in an on-demand fashion, as and when it is required. In this case, motion estimation block 760 may be implemented so as to perform block-matching as a one-step process, in which case a motion vector with the desired sub-pixel resolution is sought directly, or it may be implemented so as to perform block-matching as a two step process. If the two-step process is adopted, the first step may comprise a search for e.g. a full or half-pixel resolution motion vector and the second step is performed in order to refine the motion vector to the desired sub-pixel resolution. As block matching is an exhaustive process, in which blocks of nxm pixels in the current frame are compared one-by-one with blocks of nxm pixels or sub-pixels in the interpolated reference frame, it should be appreciated that a sub-pixel calculated in an on-demand fashion by the on-demand pixel interpolation block 750 may need to be calculated multiple times as successive difference values are determined. In a video encoder, this approach is not the most efficient possible in terms of computational complexity/burden.

In embodiments of the encoder which use only before-hand sub-pixel value interpolation, block-matching may be performed as a one step process, as all sub-pixel values of the reference frame required to determine a motion vector with the desired sub-pixel resolution are calculated before-hand in block 730 and stored in frame store 740. Thus, they are directly available for use in the block-matching process and can be retrieved as required from frame store 740 by motion estimation block 760. However, even in the case where all sub-pixel values are available from frame store 740, it is still more computationally efficient to perform block-matching as a two-step process, as fewer difference calculations are required. It should be appreciated that while full before-hand sub-pixel value interpolation reduces computational complexity in the encoder, it is not the most efficient approach in terms of memory consumption.

In embodiments of the encoder in which both before-hand and on-demand sub-pixel value interpolation are used, motion estimation block 760 is implemented in such a way that it can retrieve sub-pixel values previously calculated in before-hand sub-pixel value interpolation block 730 and stored in frame store 740 and further control on-demand sub-pixel value interpolation block 750 to calculate any additional sub-pixel values that may be required. The block-matching process may be performed as a one-step or a two-step process. If a two-step implementation is used, before-hand calculated sub-pixel values retrieved from frame store 740 may be used in the first step of the process and the second step may be implemented so as to use sub-pixel values calculated by on-demand sub-pixel value interpolation block 750. In this case, certain sub-pixel values used in the second step of the block matching process may need to be calculated multiple times as successive comparisons are made, but the number of such duplicate calculations is significantly less than if before-hand sub-pixel value calculation is not used. Furthermore, memory consumption is reduced with respect to embodiments in which only before-hand sub-pixel value interpolation is used.

Once the motion estimation block 760 has produced a motion vector for the macroblock of the current frame under examination, it outputs the motion vector to the motion field coding block 770. Motion field coding block 770 then approximates the motion vector received from motion estimation block 760 using a motion model. The motion model generally comprises a set of basis functions. More

specifically, the motion field coding block **770** represents the motion vector as a set of coefficient values (known as motion coefficients) which, when multiplied by the basis functions, form an approximation of the motion vector. The motion coefficients **724** are passed from motion field coding block **770** to motion compensated prediction block **780**. Motion compensated prediction block **780** also receives the pixel/sub-pixel values of the best-matching candidate test region of the reference frame identified by motion estimation block **760**. In FIG. 7, these values are shown to be passed via line **729** from on-demand sub-pixel interpolation block **750**. In alternative embodiments of the invention, the pixel values in question are provided from the motion estimation block **760** itself.

Using the approximate representation of the motion vector generated by motion field coding block **770** and the pixel/sub-pixel values of the best-matching candidate test region, motion compensated prediction block **780** produces a macroblock of predicted pixel values. The macroblock of predicted pixel values represents a prediction for the pixel values of the current macroblock generated from the interpolated reference frame. The macroblock of predicted pixel values is passed to the combiner **716** where it is subtracted from the new current frame in order to produce prediction error information **723** for the macroblock, as described in the foregoing.

The motion coefficients **724** formed by motion field coding block are also passed to the MUX/DMUX unit **790**, where they are combined with prediction error information **723** for the macroblock in question and possible control information from control manager **720** to form an encoded video stream **725** for transmission to a receiving terminal.

Operation of a video decoder **800** according to the invention will now be described. Referring to FIG. 8, the decoder **800** comprises a demultiplexing unit (MUX/DMUX) **810**, which receives the encoded video stream **725** from the encoder **700** and demultiplexes it, an inverse quantiser **820**, an inverse DCT transformer **830**, a motion compensated prediction block **840**, a frame store **850**, a combiner **860**, a control manager **870**, an output **880**, before-hand sub-pixel value interpolation block **845** and on-demand sub-pixel interpolation block **890** associated with the motion compensated prediction block **840**. In practice the control manager **870** of the decoder **800** and the control manager **720** of the encoder **700** may be the same processor. This may be the case if the encoder **700** and decoder **800** are part of the same video codec.

FIG. 8 shows an embodiment in which a combination of before-hand and on-demand sub-pixel value interpolation is used in the decoder. In other embodiments, only before-hand sub-pixel value interpolation is used, in which case decoder **800** does not include on-demand sub-pixel value interpolation block **890**. In a preferred embodiment of the invention, no before-hand sub-pixel value interpolation is used in the decoder and therefore before-hand sub-pixel value interpolation block **845** is omitted from the decoder architecture. If both before-hand and on-demand sub-pixel value interpolation are performed, the decoder comprises both blocks **845** and **890**.

The control manager **870** controls the operation of the decoder **800** in response to whether an INTRA or an INTER frame is being decoded. An INTRA/INTER trigger control signal, which causes the decoder to switch between decoding modes is derived, for example, from picture type information provided in the header portion of each compressed video frame received from the encoder. The INTRA/INTER

trigger control signal is passed to control manager **870** via control line **815**, together with other video codec control signals demultiplexed from the encoded video stream **725** by the MUX/DMUX unit **810**.

When an INTRA frame is decoded, the encoded video stream **725** is demultiplexed into INTRA coded macroblocks and control information. No motion vectors are included in the encoded video stream **725** for an INTRA coded frame. The decoding process is performed macroblock-by-macroblock. When the encoded information **723** for a macroblock is extracted from video stream **725** by MUX/DMUX unit **810**, it is passed to inverse quantiser **820**. The control manager controls inverse quantiser **820** to apply a suitable level of inverse quantisation to the macroblock of encoded information, according to control information provided in video stream **725**. The inverse quantised macroblock is then inversely transformed in the inverse DCT transformer **830** to form a decoded block of image information. Control manager **870** controls combiner **860** to prevent any reference information being used in the decoding of the INTRA coded macroblock. The decoded block of image information is passed to the video output **880** of the decoder.

In embodiments of the decoder which employ before-hand sub-pixel value interpolation, the decoded block of image information (i.e. pixel values) produced as a result of the inverse quantisation and inverse transform operations performed in blocks **820** and **830** is passed to before-hand sub-pixel value interpolation block **845**. Here, sub-pixel value interpolation is performed according to the method of the invention, the degree of before-hand sub-pixel value interpolation applied being determined by the details of the decoder implementation. In embodiments of the invention in which on-demand sub-pixel value interpolation is not performed, before-hand sub-pixel value interpolation block **845** interpolates all sub-pixel values. In embodiments that use a combination of before-hand and on-demand sub-pixel value interpolation, before-hand sub-pixel value interpolation block **845** interpolates a certain sub-set of sub-pixel values. This may comprise, for example, all sub-pixels at half-pixel locations, or a combination of sub-pixels at half-pixel and one quarter-pixel locations. In any case, after before-hand sub-pixel value interpolation, the interpolated sub-pixel values are stored in frame store **850**, together with the original decoded pixel values. As subsequent macroblocks are decoded, before-hand interpolated and stored, a decoded frame, at least partially interpolated to sub-pixel resolution is progressively assembled in the frame store **850** and becomes available for use as a reference frame for motion compensated prediction.

In embodiments of the decoder which do not employ before-hand sub-pixel value interpolation, the decoded block of image information (i.e. pixel values) produced as a result of the inverse quantisation and inverse transform operations performed on the macroblock in blocks **820** and **830** is passed directly to frame store **850**. As subsequent macroblocks are decoded and stored, a decoded frame, having unit pixel resolution is progressively assembled in the frame store **850** and becomes available for use as a reference frame for motion compensated prediction.

When an INTER frame is decoded, the encoded video stream **725** is demultiplexed into encoded prediction error information **723** for each macroblock of the frame, associated motion coefficients **724** and control information. Again, the decoding process is performed macroblock-by-macroblock. When the encoded prediction error information **723** for a macroblock is extracted from the video stream **725** by MUX/DMUX unit **810**, it is passed to inverse quantiser

820. Control manager **870** controls inverse quantiser **820** to apply a suitable level of inverse quantisation to the macroblock of encoded prediction error information, according to control information received in video stream **725**. The inverse quantised macroblock of prediction error information is then inversely transformed in the inverse DCT transformer **830** to yield decoded prediction error information for the macroblock.

The motion coefficients **724** associated with the macroblock in question are extracted from the video stream **725** by MUX/DMUX unit **810** and passed to motion compensated prediction block **840**, which reconstructs a motion vector for the macroblock using the same motion model as that used to encode the INTER-coded macroblock in encoder **700**. The reconstructed motion vector approximates the motion vector originally determined by motion estimation block **760** of the encoder. The motion compensated prediction block **840** of the decoder uses the reconstructed motion vector to identify the location of a block of pixel/sub-pixel values in a prediction reference frame stored in frame store **850**. The reference frame may be, for example, a previously decoded INTRA frame, or a previously decoded INTER frame. In either case, the block of pixel/sub-pixel values indicated by the reconstructed motion vector, represents the prediction of the macroblock in question.

The reconstructed motion vector may point to any pixel or sub-pixel. If the motion vector indicates that the prediction for the current macroblock is formed from pixel values (i.e. the values of pixels at unit pixel locations), these can simply be retrieved from frame store **850**, as the values in question are obtained directly during the decoding of each frame. If the motion vector indicates that the prediction for the current macroblock is formed from sub-pixel values, these must either be retrieved from frame store **850**, or calculated in on-demand sub-pixel interpolation block **890**. Whether sub-pixel values must be calculated, or can simply be retrieved from the frame store, depends on the degree of before-hand sub-pixel value interpolation used in the decoder.

In embodiments of the decoder that do not employ before-hand sub-pixel value interpolation, the required sub-pixel values are all calculated in on-demand sub-pixel value interpolation block **890**. On the other hand, in embodiments in which all sub-pixel values are interpolated before-hand, motion compensated prediction block **840** can retrieve the required sub-pixel values directly from the frame store **850**. In embodiments that use a combination before-hand and on-demand sub-pixel value interpolation, the action required in order to obtain the required sub-pixel values depends on which sub-pixel values are interpolated before-hand. Taking as an example an embodiment in which all sub-pixel values at half-pixel locations are calculated before-hand, it is evident that if a reconstructed motion vector for a macroblock points to a pixel at unit location or a sub-pixel at half-pixel location, all the pixel or sub-pixel values required to form the prediction for the macroblock are present in the frame store **850** and can be retrieved from there by motion compensated prediction block **840**. If, however, the motion vector indicates a sub-pixel at a quarter-pixel location, the sub-pixels required to form the prediction for the macroblock are not present in frame store **850** and are therefore calculated in on-demand sub-pixel value interpolation block **890**. In this case, on-demand sub-pixel value interpolation block **890** retrieves any pixel or sub-pixel required to perform the interpolation from frame store **850** and applies the interpolation method described below. Sub-pixel values calculated in on-demand sub-pixel value interpolation block **890** are passed to motion compensated prediction block **840**.

Once a prediction for a macroblock has been obtained, the prediction (that is, a macroblock of predicted pixel values) is passed from motion compensated prediction block **840** to combiner **860** where it combined with the decoded prediction error information for the macroblock to form a reconstructed image block which, in turn, is passed to the video output **880** of the decoder.

It should be appreciated that in practical implementations of encoder **700** and decoder **800**, the extent to which frames are before-hand sub-pixel interpolated, and thus the amount of on-demand sub-pixel value interpolation that is performed, can be chosen according to, or dictated by, the hardware implementation of the video encoder **700**, or the environment in which it is intended to be used. For example, if the memory available to the video encoder is limited, or memory must be reserved for other functions, it is appropriate to limit the amount of before-hand sub-pixel value interpolation that is performed. In other cases, where the microprocessor performing the video encoding operation has limited processing capacity, e.g. the number of operations per second that can be executed is comparatively low, it is more appropriate to restrict the amount of on-demand sub-pixel value interpolation that is performed. In a mobile communications environment, for example, when video encoding and decoding functionality is incorporated in a mobile telephone or similar wireless terminal for communication with a mobile telephone network, both memory and processing power may be limited. In this case a combination of before-hand and on-demand sub-pixel value interpolation may be the best choice to obtain an efficient implementation in the video encoder. In video decoder **800**, use of before-hand sub-pixel value is generally not preferred, as it typically results in the calculation of many sub-pixel values that are not actually used in the decoding process. However, it should be appreciated that although different amounts of before-hand and on-demand interpolation can be used in the encoder and decoder in order to optimise the operation of each, both encoder and decoder can be implemented so as to use the same division between before-hand and on-demand sub-pixel value interpolation.

Although the foregoing description does not describe the construction of bi-directionally predicted frames (B-frames) in the encoder **700** and the decoder **800**, it should be understood that in embodiments of the invention, such a capability may be provided. Provision of such capability is considered within the ability of one skilled in the art.

An encoder **700** or a decoder **800** according to the invention can be realised using hardware or software, or using a suitable combination of both. An encoder or decoder implemented in software may be, for example, a separate program or a software building block that can be used by various programs. In the above description and in the drawings, the functional blocks are represented as separate units, but the functionality of these blocks can be implemented, for example, in one software program unit.

The encoder **700** and decoder **800** can further be combined in order to form a video codec having both encoding and decoding functionality. In addition to being implemented in a multimedia terminal, such a codec may also be implemented in a network. A codes according to the invention may be a computer program or a computer program element, or it may be implemented at least partly using hardware.

The sub-pixel interpolation method used in the encoder **700** and decoder **800** according to the invention now be described in detail. The method will first be introduced at a

general conceptual level and then two preferred embodiments will be described. In the first preferred embodiment, sub-pixel value interpolation is performed to $\frac{1}{4}$ pixel resolution and in the second the method is extended to $\frac{1}{8}$ pixel resolution.

It should be noted that interpolation must produce identical values in the encoder and the decoder, but its implementation should be optimized for both entities separately. For example, in an encoder according to the first embodiment of the invention in which sub-pixel value interpolation is performed to $\frac{1}{4}$ pixel resolution, it is most efficient to calculate $\frac{1}{2}$ resolution pixels before-hand and to calculate values for $\frac{1}{4}$ resolution sub-pixels in an on-demand fashion, only when they are needed during motion estimation. This has the effect of limiting memory usage while keeping the computational complexity/burden at an acceptable level. In the decoder, on the other hand, it is advantageous not to pre-calculate any of the sub-pixels. Therefore, it should be appreciated that a preferred embodiment of the decoder does not include before-hand sub-pixel value interpolation block 845 and all sub-pixel value interpolation is performed in on-demand sub-pixel value interpolation block 890.

In the description of the interpolation method provided below, references are made to the pixel positions depicted in FIG. 14a. In this figure pixels labelled A represent original pixels (that is, pixels residing at unit horizontal and vertical locations). Pixels labelled with other letters represent sub-pixels that are to be interpolated. The description that follows will adhere to the previously introduced conventions regarding the description of pixel and sub-pixel locations.

Next, the steps required to interpolate all sub-pixel positions are described:

Values for the $\frac{1}{2}$ resolution sub-pixels labelled b are obtained by first calculating an intermediate value b using a Kth order filter, according to:

$$b = \sum_{i=1}^K x_i A_i \quad (9)$$

where x_i is a vector of filter coefficients, A_i is a corresponding vector of original pixel values A situated at unit horizontal and unit vertical locations, and K is an integer which defines the order of the filter. Thus, equation 9 can be re-expressed as:

$$b = x_1 A_1 + x_2 A_2 + x_3 A_3 + \dots + x_{K-1} A_{K-1} + x_K A_K \quad (10)$$

The values of the filter coefficients x_i and the order of the filter K may vary from embodiment to embodiment. Equally, different coefficient values may be used in the calculation of different sub-pixels within an embodiment. In other embodiments, the values of filter coefficients x_i and the order of the filter may depend on which of the $\frac{1}{2}$ resolution b sub-pixels is being interpolated. Pixels A_i are disposed symmetrically with respect to the $\frac{1}{2}$ resolution sub-pixel b being interpolated and are the closest neighbours of that sub-pixel. In the case of the $\frac{1}{2}$ resolution sub-pixel b situated at half horizontal and unit vertical location, pixels A_i are disposed horizontally with respect to b (as shown in FIG. 14b). If the $\frac{1}{2}$ resolution sub-pixel b situated at unit horizontal and half vertical location is being interpolated, pixels A_i are disposed vertically with respect to b (as shown in FIG. 14c).

A final value for $\frac{1}{2}$ resolution sub-pixel b is calculated by dividing intermediate value b by a constant scale₁, truncating it to obtain an integer number and clipping the result to lie

in the range $[0, 2^n - 1]$. In alternative embodiments of the invention rounding may be performed instead of truncation. Preferably, constant scale, is chosen to be equal to the sum of filter coefficients x_i .

A value for the $\frac{1}{2}$ resolution sub-pixel labelled c is also obtained by first calculating an intermediate value c using an Mth order filter, according to:

$$c = \sum_{i=1}^M y_i b_i \quad (11)$$

where y_i is a vector of filter coefficients, b_i is a corresponding vector of intermediate values b_i in the horizontal or vertical direction. i.e.:

$$c = y_1 b_1 + y_2 b_2 + y_3 b_3 + \dots + y_{M-1} b_{M-1} + y_M b_M \quad (12)$$

The values of the filter coefficients y_i and the order of the filter M may vary from embodiment to embodiment. Equally, different coefficient values may be used in the calculation of different sub-pixels within an embodiment. Preferably, the b values are intermediate values for $\frac{1}{2}$ resolution sub-pixels b which are disposed symmetrically with respect to $\frac{1}{2}$ resolution sub-pixel c and are the closest neighbours of sub-pixel c. In an embodiment of the invention, the $\frac{1}{2}$ resolution sub-pixels b are disposed horizontally with respect to sub-pixel c, in an alternative embodiment they are disposed vertically with respect to sub-pixel c.

A final value of $\frac{1}{2}$ resolution sub-pixel c is computed by dividing intermediate value c by a constant scale₂, truncating it to obtain an integer number and clipping the result to lie in the range $[0, 2^n - 1]$. In alternative embodiments of the invention rounding may be performed instead of truncation. Preferably, constant scale₂ is equal to scale₁ * scale₁.

It should be noted that the use of intermediate values b in the horizontal direction leads to the same result as using intermediate values b in the vertical direction.

There are two alternatives for interpolating values for the $\frac{1}{4}$ resolution sub-pixels labelled h. Both involve linear interpolation along a diagonal line linking $\frac{1}{2}$ resolution sub-pixels neighbouring the $\frac{1}{4}$ resolution sub-pixel h being interpolated. In a first embodiment, a value for sub-pixel h is calculated by averaging the values of the two $\frac{1}{2}$ resolution sub-pixels b closest to sub-pixel h. In a second embodiment, a value for sub-pixel h is calculated by averaging the values of the closest pixel A and the closest $\frac{1}{2}$ resolution sub-pixel c. It should be appreciated that this provides the possibility of using different combinations of diagonal interpolations to determine the values for sub-pixels h within the confines of different groups of 4 image pixels A. However, it should also be realised that the same combination should be used in both the encoder and the decoder in order to produce identical interpolation results. FIG. 15 depicts 4 possible choices of diagonal interpolation for sub-pixels h in adjacent groups of 4 pixels within an image. Simulations in the TML environment have verified that both embodiments result in similar compression efficiency. The second embodiment has higher complexity, since calculation of sub-pixel c requires calculation of several intermediate values. Therefore the first embodiment is preferred.

Values for $\frac{1}{4}$ resolution sub-pixels labelled d and g are calculated from the values of their nearest horizontal neighbours using linear interpolation. In other words, a value for $\frac{1}{4}$ resolution sub-pixel d is obtained by averaging values of its nearest horizontal neighbours, original image pixel A and $\frac{1}{2}$ resolution sub-pixel b. Similarly, a value for $\frac{1}{4}$ resolution

37

sub-pixel g is obtained by taking the average of its two nearest horizontal neighbours, $\frac{1}{2}$ resolution sub-pixels b and c.

Values for $\frac{1}{4}$ resolution sub-pixels labelled e, f and i are calculated from the values of their nearest neighbours in the vertical direction using linear interpolation. More specifically, a value for $\frac{1}{4}$ resolution sub-pixel e is obtained by averaging the values of its two nearest vertical neighbours, original image pixel A and $\frac{1}{2}$ resolution sub-pixel b. Similarly, a value for $\frac{1}{4}$ resolution sub-pixel f is obtained by taking the average of its two nearest vertical neighbours, $\frac{1}{2}$ resolution sub-pixels b and c. In an embodiment of the invention, a value for $\frac{1}{4}$ resolution sub-pixel i is obtained in manner identical to that just described in connection with $\frac{1}{4}$ resolution sub-pixel f. However, in an alternative embodiment of the invention, and in common with the H.26 test models TML5 and TML6 previously described, $\frac{1}{4}$ resolution sub-pixel i is determined using the values of the four closest original image pixels, according to $(A_1 + A_2 + A_3 + A_4)/4$.

It should also be noted that in all cases where an average involving pixel and/or sub-pixel values is determined, the average may be formed in any appropriate manner. For example, the value for $\frac{1}{4}$ resolution sub-pixel d can be defined as $d = (A + b)/2$ or as $d = (A + b + 1)/2$. The addition of 1 to the sum of values for pixel A and $\frac{1}{2}$ resolution sub-pixel b has the effect of causing any rounding or truncation operation subsequently applied to round or truncate the value for d to the next highest integer value. This is true for any sum of integer values and may be applied to any of the averaging operations performed according to the method of the invention in order to control rounding or truncation effects.

It should be noted that the sub-pixel value interpolation method according to the invention provides advantages over each of TML5 and TML6.

In contrast to TML5, in which the values of some of the $\frac{1}{4}$ resolution sub-pixels depend on previously interpolated values obtained for other $\frac{1}{4}$ resolution sub-pixels, in the method according to the invention, all $\frac{1}{4}$ resolution sub-pixels are calculated from original image pixels or $\frac{1}{2}$ resolution sub-pixel positions using linear interpolation. Thus, the reduction in precision of those $\frac{1}{4}$ resolution sub-pixel values which occurs in TML5 due to the intermediate truncation and clipping of the other $\frac{1}{4}$ resolution sub-pixels from which they are calculated, does not take place in the method according to the invention. In particular, referring to FIG. 14a, $\frac{1}{4}$ resolution sub-pixels h (and sub-pixel i in one embodiment of the invention) are interpolated diagonally in order to reduce dependency on other $\frac{1}{4}$ -pixels. Furthermore, in the method according to the invention, the number of calculations (and therefore the number of processor cycles) required to obtain a value for those $\frac{1}{4}$ resolution sub-pixels in the decoder is reduced compared with TML5. Additionally, the calculation of any $\frac{1}{4}$ resolution sub-pixel value requires a number of calculations which is substantially similar to the number of calculations required to determine any other $\frac{1}{4}$ resolution sub-pixel value. More specifically, in a situation where the required $\frac{1}{2}$ resolution sub-pixel values are already available, e.g. they have been calculated before-hand, the number of calculations required to interpolate a $\frac{1}{4}$ resolution sub-pixel value from the pre-calculated $\frac{1}{2}$ resolution sub-pixel values is the same as the number of calculations required to calculate any other $\frac{1}{4}$ resolution sub-pixel value from the available $\frac{1}{2}$ resolution sub-pixel values.

In comparison with TML6, the method according to the invention does not require high precision arithmetic to be

38

used in the calculation of all sub-pixels. Specifically, as all of the $\frac{1}{4}$ resolution sub-pixel values are calculated from original image pixels or $\frac{1}{2}$ resolution sub-pixel values using linear interpolation, lower precision arithmetic can be used in their interpolation. Consequently, in hardware implementations of the inventive method, for example in an ASIC (Application Specific Integrated Circuit), the use of lower precision arithmetic reduces the number of components (e.g. gates) that must be devoted to the calculation of $\frac{1}{4}$ resolution sub-pixel values. This, in turn, reduces the overall area of silicon that must be dedicated to the interpolation function. As the majority of sub-pixels are, in fact, $\frac{1}{4}$ resolution sub-pixels (12 out of the 15 sub-pixels illustrated in FIG. 14a), the advantage provided by the invention in this respect is particularly significant. In software implementations, where sub-pixel interpolation is performed using the standard instruction set of a general purpose CPU (Central Processor Unit) or using a DSP (Digital Signal Processor), a reduction in the precision of the arithmetic required generally leads to an increase in the speed at which the calculations can be performed. This is particularly advantageous in 'low cost' implementations, in which it is desirable to use a general purpose CPU rather than any form of ASIC.

The method according to the invention provides still further advantages compared with TML5. As mentioned previously, in the decoder only 1 out of the 15 sub-pixel positions is required at any given time, namely that which is indicated by received motion vector information. Therefore, it is advantageous if the value of a sub-pixel in any sub-pixel location can be calculated with the minimum number of steps that result in a correctly interpolated value. The method according to the invention provides this capability. As mentioned in the detailed description provided above, $\frac{1}{2}$ resolution sub-pixel c can be interpolated by filtering in either the vertical or the horizontal direction, the same value being obtained for c regardless of whether horizontal or vertical filtering is used. The decoder can therefore take advantage of this property when calculating values for $\frac{1}{4}$ resolution sub-pixels f and g in such a way as to minimise the number of operations required in order to obtain the required values. For example, if the decoder requires a value for $\frac{1}{4}$ resolution sub-pixel f, $\frac{1}{2}$ resolution sub-pixel c should be interpolated in the vertical direction. If a value is required for $\frac{1}{4}$ resolution sub-pixel g, it is advantageous to interpolate a value for c in the horizontal direction. Thus, in general, it can be said that the method according to the invention provides flexibility in the way in which values are derived for certain $\frac{1}{4}$ resolution sub-pixels. No such flexibility is provided in TML5.

Two specific embodiments will now be described in detail. The first represents a preferred embodiment for calculating sub-pixels with up to $\frac{1}{4}$ pixel resolution, while in the second, the method according to the invention is extended to the calculation of values for sub-pixels having up to $\frac{1}{8}$ pixel resolution. For both embodiments a comparison is provided between the computational complexity/burden resulting from use of the method according to the invention and that which would result from use of the interpolation methods according to TML5 and TML6 in equivalent circumstances.

The preferred embodiment for interpolating sub-pixels at $\frac{1}{4}$ pixel resolution will be described with reference to FIGS. 14a, 14b and 14c. In the following, it will be assumed that all image pixels and final interpolated values for sub-pixels are represented with 8-bits.

Calculation of $\frac{1}{2}$ resolution sub-pixels at i) half unit horizontal and unit vertical locations and ii) unit horizontal and half unit vertical locations.

1. A value for the sub-pixel at half unit horizontal and unit vertical location, that is $\frac{1}{2}$ resolution sub-pixel b in FIG. 14a, is obtained by first calculating intermediate value $b=(A_1-5A_2+20A_3+20A_4-5A_5+A_6)$ using the values of the six pixels (A_1 to A_6) which are situated at unit horizontal and unit vertical locations in either the row or the column of pixels containing b and which are disposed symmetrically about b, as shown in FIGS. 14b and 14c. A final value for $\frac{1}{2}$ resolution sub-pixel b is calculated as $(b+16)/32$ where the operator / denotes division with truncation. The result is clipped to lie in the range [0, 255].

Calculation of $\frac{1}{2}$ resolution sub-pixels at half unit horizontal and half unit vertical locations.

2. A value for the sub-pixel at half unit horizontal and half unit vertical location, that is $\frac{1}{2}$ resolution sub-pixel c in FIG. 14a, is calculated as $c=(b_1-5b_2+20b_3+20b_4-5b_5+b_6+512)/1024$ using the intermediate values b for the six closest $\frac{1}{2}$ resolution sub-pixels which are situated in either the row or the column of sub-pixels containing c and which are disposed symmetrically about c. Again, operator / denotes division with truncation and the result is clipped to lie in the range [0, 255]. As previously explained, using intermediate values b for $\frac{1}{2}$ resolution sub-pixels b in the horizontal direction leads to the same result as using intermediate values b for $\frac{1}{2}$ resolution sub-pixels b in the vertical direction. Thus, in an encoder according to the invention, the direction for interpolating $\frac{1}{2}$ resolution sub-pixels b can be chosen according to a preferred mode of implementation. In a decoder according to the invention, the direction for interpolating sub-pixels b is chosen according to which, if any, $\frac{1}{4}$ resolution sub-pixels will be interpolated using the result obtained for $\frac{1}{2}$ resolution sub-pixel c.

Calculation of $\frac{1}{4}$ resolution sub-pixels at i) quarter unit horizontal and unit vertical locations; ii) quarter unit horizontal and half unit vertical locations; iii) unit horizontal and quarter unit vertical locations; and iv) half unit horizontal and quarter unit vertical locations.

3. Values for $\frac{1}{4}$ resolution sub-pixels d, situated at quarter unit horizontal and unit vertical locations are calculated according to $d=(A+b)/2$ using the nearest original image pixel A and the closest $\frac{1}{2}$ resolution sub-pixel b in the horizontal direction. Similarly, values for $\frac{1}{4}$ resolution sub-pixels g, situated at quarter unit horizontal and half unit vertical locations are calculated according to $g=(b+c)/2$ using the two nearest $\frac{1}{2}$ resolution sub-pixels in the horizontal direction. In a similar manner, values for $\frac{1}{4}$ resolution sub-pixels e, situated at unit horizontal and quarter unit vertical locations, are calculated according to $e=(A+b)/2$ using the nearest original image pixel A and the closest $\frac{1}{2}$ resolution sub-pixel b in the vertical direction. Values for $\frac{1}{4}$ resolution sub-pixels f, situated at half unit horizontal and quarter unit vertical locations, are determined from $f=(b+c)/2$ using the two nearest $\frac{1}{2}$ resolution sub-pixel in the vertical direction. In all cases, operator / denotes division with truncation.

Calculation of $\frac{1}{4}$ resolution sub-pixels at quarter unit horizontal and quarter unit vertical locations.

4. Values for $\frac{1}{4}$ resolution sub-pixels h, situated at quarter unit horizontal and quarter unit vertical locations are calculated according to $h=(b_1+b_2)/2$, using the two nearest $\frac{1}{2}$ resolution sub-pixels b in the diagonal direction. Again, operator / denotes division with truncation.

5. A value for the $\frac{1}{4}$ resolution sub-pixel labeled i is computed from $i=(A_1+A_2+A_3+A_4+2)/4$ using the four nearest original pixels A. Once more, operator / denotes division with truncation.

An analysis of the computational complexity of the first preferred embodiment of the invention will now be presented.

In the encoder, it is likely that the same sub-pixel values will be calculated multiple times. Therefore, and as previously explained, the complexity of the encoder can be reduced by pre-calculating all sub-pixel values and storing them in memory. However, this solution increases memory usage by a large margin. In the preferred embodiment of the invention, in which motion vector accuracy is $\frac{1}{4}$ pixel resolution in both the horizontal and vertical dimensions, storing pre-calculated sub-pixel values for the whole image requires 16 times the memory required to store the original, non-interpolated image. To reduce memory usage, all $\frac{1}{2}$ resolution sub-pixels can be interpolated before-hand and $\frac{1}{4}$ resolution sub-pixels can be calculated on-demand, that is, only when they are needed. According to the method of the invention, on-demand interpolation of values for $\frac{1}{4}$ resolution sub-pixels only requires linear interpolation from $\frac{1}{2}$ resolution sub-pixels. Four times the original picture memory is required to store the pre-calculated $\frac{1}{2}$ resolution sub-pixels since only 8 bits are necessary to represent them.

However, if the same strategy of pre-calculating all $\frac{1}{2}$ resolution sub-pixels using before-hand interpolation is used in conjunction with the direct interpolation scheme of TML6, the memory requirements increase to 9 times the memory required to store the original non-interpolated image. This results from the fact that a larger number of bits is required to store the high precision intermediate values associated with each $\frac{1}{2}$ resolution sub-pixel in TML6. In addition, the complexity of sub-pixel interpolation during motion estimation is higher in TML6, since scaling and clipping has to be performed for every $\frac{1}{2}$ and $\frac{1}{4}$ sub-pixel location.

In the following, the complexity of the sub-pixel value interpolation method according to the invention, when applied in a video decoder, is compared with that of the interpolation schemes used in TML5 and TML6. Throughout the analysis which follows, it is assumed that in each method the interpolation of any sub-pixel value is performed using only the minimum number of steps required to obtain a correctly interpolated value. It is further assumed that each method is implemented in a block based manner, that is, intermediate values common for all the sub-pixels to be interpolated in a particular $N \times M$ block are calculated only once. An illustrative example is provided in FIG. 16. Referring to FIG. 16, it can be seen that in order to calculate a 4×4 block of $\frac{1}{2}$ resolution sub-pixels c, a 9×4 block of $\frac{1}{2}$ resolution sub-pixels b is first calculated.

Compared with the sub-pixel value interpolation method used in TML5, the method according to the invention has a lower computational complexity for the following reasons:

1. Unlike the sub-pixel value interpolation scheme used in TML5, according to the method of the invention, a value for $\frac{1}{2}$ resolution sub-pixel c can be obtained by filtering in either the vertical or the horizontal direction. Thus, in order to reduce the number of operations, $\frac{1}{2}$ resolution sub-pixel c can be interpolated in the vertical direction if a value for $\frac{1}{4}$ resolution sub-pixel f is required and in the horizontal direction if a value for $\frac{1}{4}$ resolution sub-pixel g is required. As an example, FIG. 17 shows all the $\frac{1}{2}$ resolution sub-pixel values that must be calculated in order to interpolate values for $\frac{1}{4}$ resolution sub-pixels g in

an image block defined by 4x4 original image pixels using the interpolation method of TML5 (FIG. 17a) and using the method according to the invention (FIG. 17b). In this example, the sub-pixel value interpolation method according to TML5 requires a total of 88½ resolution sub-pixels to be interpolated, while the method according to the invention requires the calculation of 72½ resolution sub-pixels. As can be seen from FIG. 17b, according to the invention, ½ resolution sub-pixels c are interpolated in the horizontal direction in order to reduce the number of calculations required.

2. According to the method of the invention, ¼ resolution sub-pixel h is calculated by linear interpolation from its two closest neighbouring ½ resolution sub-pixels in the diagonal direction. The respective numbers of ½ resolution sub-pixels that must be calculated in order to obtain values for ¼ resolution sub-pixels h within a 4x4 block of original image pixels using the sub-pixel value interpolation method according to TML5 and the method according to the invention are shown in FIGS. 18(a) and 18(b), respectively. Using the method according to TML5 it is necessary to interpolate a total of 56½ resolution sub-pixels, while according to the method of the invention it is necessary to interpolate 40½ resolution sub-pixels.

Table 1 summarizes the decoder complexities of the three sub-pixel value interpolation methods considered here, that according to TML5, the direct interpolation method used in TML6 and the method according to the invention. Complexity is measured in terms of the number of 6-tap filtering and linear interpolation operations performed. It is assumed that Interpolation of ¼ resolution sub-pixel i is calculated according to $i=(A_1+A_2+A_3+A_4+2)/4$ which is bilinear interpolation and effectively comprises two linear interpolation operations. The operations needed to interpolate sub-pixel values with one 4x4 block of original image pixels are listed for each of the 15 sub-pixel positions which, for convenience of reference, are numbered according to the scheme shown in FIG. 19. Referring to FIG. 19, location 1 is the location of an original image pixel A and locations 2 to 16 are sub-pixel locations. Location 16 is the location of ¼ resolution sub-pixel i. In order to compute the average number of operations it has been assumed that the probability of a motion vector pointing to each sub-pixel position is the same. The average complexity is therefore the average of the 15 sums calculated for each sub-pixel location and the single full-pixel location.

TABLE 1

Complexity of ¼ resolution sub-pixel interpolation in TML5, TML6 and the method according to the invention.						
Location	TML5		TML6		Inventive Method	
	linear.	6-tap	linear.	6-tap	linear.	6-tap
1	0	0	0	0	0	0
3, 9	0	16	0	16	0	16
2, 4, 5, 13	16	16	0	16	16	16
11	0	52	0	52	0	52
7, 15	16	52	0	52	16	52
10, 12	16	68	0	52	16	52
6, 8, 14	48	68	0	52	16	32
16	32	0	32	0	32	0
Average	19	37	2	32	13	28.25

It can be seen from Table 1 that the method according to the invention requires fewer 6-tap filter operations than the sub-pixel value interpolation method according to TML6 and only a few additional linear interpolation operations.

Since 6-tap filter operations are much more complex than linear interpolation operations the complexity of the two methods is similar. The sub-pixel value interpolation method according to TML5 has a considerably higher complexity.

The preferred embodiment for interpolating sub-pixels with up to ⅛ pixel resolution will now be described with reference to FIGS. 20, 21 and 22.

FIG. 20 presents the nomenclature used to describe pixels, ½ resolution sub-pixels, ¼ resolution sub-pixels and ⅛ resolution sub-pixels in this extended application of the method according to the invention.

1. Values for the ½ resolution and ¼ resolution sub-pixels labelled b^1 , b^2 and b^3 in FIG. 20 are obtained by first calculating intermediate values $b^1=(-3A_1+12A_2-37A_3+229A_4+71A_5-21A_6+6A_7-A_8)$; $b^2=(-3A_1+12A_2-39A_3+158A_4+158A_5-39A_6+12A_7-3A_8)$; and $b^3=(-A_1+6A_2-21A_3+71A_4+229A_5-37A_6+13A_7-3A_8)$ using the values of the eight nearest image pixels (A_1 to A_8) situated at unit horizontal and unit vertical locations in either the row or the column containing b^1 , b^2 and b^3 and disposed symmetrically about ½ resolution sub-pixel b^2 . The asymmetries in the filter coefficients used to obtain intermediate values b^1 and b^3 account for the fact that pixels A_1 to A_8 are not symmetrically located with respect to ¼ resolution sub-pixels b^1 and b^3 . Final values for sub-pixels b^i , $i=1, 2, 3$ are calculated according to $b^i=(b^i+128)/256$ where the operator / denotes division with truncation. The result is clipped to lie in the range [0, 255].

2. Values for the ½ resolution and ¼ resolution sub-pixels labelled c^{ij} , $i, j=1, 2, 3$, are calculated according to $c^{1j}=(-3b_1^j+12b_2^j-37b_3^j+229b_4^j+71b_5^j-21b_6^j+6b_7^j-b_8^j+32768)/65536$, $c^{2j}=(-3b_1^j+12b_2^j-39b_3^j+158b_4^j+158b_5^j-39b_6^j+12b_7^j-3b_8^j+32768)/65536$ and $c^{3j}=(-b_1^j+6b_2^j-21b_3^j+71b_4^j+229b_5^j-37b_6^j+13b_7^j-3b_8^j+32768)/65536$ using the intermediate values b^1 , b^2 and b^3 calculated for the eight closest sub-pixels (b_1^j to b_8^j) in the vertical direction, sub-pixels b^j being situated in the column comprising the ½ resolution and ¼ resolution sub-pixels c^{ij} being interpolated and disposed symmetrically about the ½ resolution sub-pixel c^{2j} . The asymmetries in the filter coefficients used to obtain values for sub-pixels c^{1j} and c^{3j} account for the fact that sub-pixels b_1^j to b_8^j are not symmetrically located with respect to ¼ resolution sub-pixels c^{1j} and c^{3j} . Once more, operator / denotes division with truncation. Before the interpolated values for sub-pixels c^{ij} are stored in the frame memory they are clipped to lie in the range [0, 255]. In an alternative embodiment of the invention, ½ resolution and ¼ resolution sub-pixels c^{ij} are calculated using in an analogous manner using intermediate values b^1 , b^2 and b^3 in the horizontal direction.

3. Values for ⅛ resolution sub-pixels labelled d are calculated using linear interpolation from the values of their closest neighbouring image pixel, ½ resolution or ¼ resolution sub-pixels in the horizontal or vertical direction. For example, upper leftmost ⅛ resolution sub-pixel d is calculated according to $d=(A+b^1+1)/2$. As before, operator / indicates division with truncation.

4. Values for ⅛ resolution sub-pixels labelled e and f are calculated using linear interpolation from the values of image pixels, ½ resolution or ¼ resolution sub-pixels in the diagonal direction. For example, referring to FIG. 20, upper leftmost pixel ⅛ resolution sub-pixel e is calculated according to $e=(b^1+b^1+1)/2$. The diagonal direction to be used in the interpolation of each ⅛ resolution sub-pixel in a first preferred embodiment of the invention, hereinafter referred to as 'preferred method 1', is indicated in FIG.

21(a). Values for $\frac{1}{8}$ resolution sub-pixels labelled g are calculated according to $g=(A+3c^{22}+3)/4$. As always, operator / denotes division with truncation. In an alternative embodiment of the invention, hereinafter referred to as 'preferred method 2', computational complexity is further reduced by interpolating $\frac{1}{8}$ resolution sub-pixels f using linear interpolation from $\frac{1}{2}$ resolution sub-pixels b^2 , that is, according to the relationship $f=(3b^2+b^2+2)/4$. The b^2 sub-pixel which is closer to f is multiplied by 3. The diagonal interpolation scheme used in this alternative embodiment of the invention is depicted in FIG. 21(b). In further alternative embodiments, different diagonal interpolation schemes can be envisaged.

It should be noted that in all cases where an average involving pixel and/or sub-pixel values is used in the determination of $\frac{1}{8}$ resolution sub-pixels, the average may be formed in any appropriate manner. The addition of 1 to the sum of values used in calculating such an average has the effect of causing any rounding or truncation operation subsequently applied to round or truncate the average in question to the next highest integer value. In alternative embodiments of the invention, the addition of 1 is not used.

As in the case of sub-pixel value interpolation to $\frac{1}{4}$ pixel resolution previously described, memory requirements in the encoder can be reduced by pre-calculating only a part of the sub-pixel values to be interpolated. In the case of sub-pixel value interpolation to $\frac{1}{8}$ pixel resolution, it is advantageous to calculate all $\frac{1}{2}$ resolution and $\frac{1}{4}$ resolution

at up to $\frac{1}{8}$ pixel resolution, is compared with that of the interpolation schemes used in TML5 and TML6. As in the equivalent analysis for $\frac{1}{4}$ pixel resolution sub-pixel value interpolation described above, it is assumed that in each method the interpolation of any sub-pixel value is performed using only the minimum number of steps required to obtain a correctly interpolated value. It is also assumed that each method is implemented in a block based manner, such that intermediate values common for all the sub-pixels to be interpolated in a particular N×M block are calculated only once.

Table 2 summarizes complexities of the three interpolation methods. Complexity is measured in terms of the number of 8-tap filter and linear interpolation operations performed in each method. The table presents the number of operations required to interpolate each of the $63\frac{1}{8}$ resolution sub-pixels within one 4×4 block of original image pixels, each sub-pixel location being identified with a corresponding number, as illustrated in FIG. 22. In FIG. 22, location 1 is the location of an original image pixel and locations 2 to 64 are sub-pixel locations. When computing the average number of operations, it has been assumed that the probability of a motion vector pointing to each sub-pixel position is the same. The average complexity is thus the average of the 63 sums calculated for each sub-pixel location and the single full-pixel location.

TABLE 2

Complexity of $\frac{1}{8}$ resolution sub-pixel interpolation in TML5, TML6 and the method according to the invention. (Results shown separately for Preferred Method 1 and Preferred Method 2).								
Location	TML5		TML6		Preferred Method 1		Preferred Method 2	
	linear.	8-tap	linear.	8-tap	linear.	8-tap	linear.	8-tap
1	0	0	0	0	0	0	0	0
3, 5, 7, 17, 33, 49	0	16	0	16	0	16	0	16
19, 21, 23, 35, 37, 39, 51, 53, 55	0	60	0	60	0	60	0	60
2, 8, 9, 57	16	16	0	16	16	16	16	16
4, 6, 25, 41	16	32	0	16	16	32	16	32
10, 16, 58, 64	32	76	0	60	16	32	16	32
11, 13, 15, 59, 61, 63	16	60	0	60	16	60	16	60
18, 24, 34, 40, 50, 56	16	76	0	60	16	60	16	60
12, 14, 60, 62	32	120	0	60	16	32	16	32
26, 32, 42, 48	32	108	0	60	16	32	16	32
20, 22, 36, 38, 52, 54	16	120	0	60	16	76	16	76
27, 29, 31, 43, 45, 47	16	76	0	60	16	76	16	76
28, 30, 44, 46	32	152	0	60	16	60	16	60
Average	64	290.25	0	197.75	48	214.75	48	192.75

sub-pixels before-hand and to compute values for $\frac{1}{8}$ resolution sub-pixels in an on-demand fashion, only when they are required. When this approach is taken, both the interpolation method according to TML5 and that according to the invention require 16 times the original picture memory to store the $\frac{1}{2}$ resolution and $\frac{1}{4}$ resolution sub-pixel values. However, if the direct interpolation method according to TML6 is used in the same way, intermediate values for the $\frac{1}{2}$ resolution and $\frac{1}{4}$ pixel resolution sub-pixels must be stored. These intermediate values are represented with 32-bit precision and this results in a memory requirement 64 times that for the original, non-interpolated image.

In the following, the complexity of the sub-pixel value interpolation method according to the invention, when applied in a video decoder to calculate values for sub-pixels

As can be seen from Table 2, the number of 8-tap filtering operations performed according to preferred methods 1 and 2 are, respectively, 26% and 34% lower than the number of 8-tap filtering operation performed in the sub-pixel value interpolation method of TML5. The number of linear operations is 25% lower, in both preferred method 1 and preferred method 2, compared with TML5, but this improvement is of lesser importance compared to the reduction in 8-tap filtering operations. It can further be seen that the direct interpolation method used in TML6 has a complexity comparable that of both preferred methods 1 and 2 when used to interpolate values for $\frac{1}{8}$ resolution sub-pixels.

In view of the foregoing description it will be evident to a person skilled in the art that various modifications may be made within the scope of the invention. While a number of

preferred embodiments of the invention have been described in detail, it should be apparent that many modifications and variations thereto are possible, all of which fall within the true spirit and scope of the invention.

What is claimed is:

1. A method of interpolation in video coding in which an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, is interpolated to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $\frac{1}{2}^x$, where x is a positive integer having a maximum value N, the method comprising:

- a) when values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations are required, interpolating such values directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
 - b) when values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations are required, interpolating such values directly using a choice of a first weighted sum of values for sub-pixels residing at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a); and
 - c) when a value for a sub-pixel situated at a $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical location is required, interpolating such a value by taking a weighted average of the value of a first sub-pixel or pixel situated at a $\frac{1}{2}^{N-m}$ unit horizontal and $\frac{1}{2}^{N-n}$ unit vertical location and the value of a second sub-pixel or pixel located at a $\frac{1}{2}^{N-p}$ unit horizontal and $\frac{1}{2}^{N-q}$ unit vertical location, variables m, n, p and q taking integer values in the range 1 to N such that the first and second sub-pixels or pixels are located diagonally with respect to the sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical location.
2. A method according to claim 1 wherein a first and a second weight are used in the weighted average referred to in (c), the relative magnitudes of the weights being inversely proportional to the (straight-line diagonal) proximity of the first and the second sub-pixel or pixel to the sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical location.

3. A method according to claim 2, wherein in a situation where the first and the second sub-pixel or pixel are symmetrically located with respect to (equidistant from) the sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical location, the first and second weights have equal values.

4. A method according to claim 1 in which the first weighted sum of values for sub-pixels residing at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations in step b) is used when a sub-pixel at $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^N$ unit vertical location is required.

5. A method according to claim 1 in which the second weighted sum of values for sub-pixels residing at unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations in step b) is used when a sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location is required.

6. A method according to claim 1 in which, when values for sub-pixels at $\frac{1}{2}^N$ unit horizontal and unit vertical locations, and $\frac{1}{2}^N$ horizontal and $\frac{1}{2}^{N-1}$ vertical locations are required, interpolating such values by taking the average of the values of a first pixel or sub-pixel located at a vertical location corresponding to that of the sub-pixel being calcu-

lated and unit horizontal location and a second pixel or sub-pixel located at a vertical location corresponding to that of the sub-pixel being calculated and $\frac{1}{2}^{N-1}$ unit horizontal location.

7. A method according to claim 1 in which, when values for sub-pixels at unit horizontal and $\frac{1}{2}^N$ unit vertical locations, and $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations are required, interpolating such values by taking the average of the values of a first pixel or sub-pixel located at a horizontal location corresponding to that of the sub-pixel being calculated and unit vertical location and a second pixel or sub-pixel located at a horizontal location corresponding to that of the sub-pixel being calculated and $\frac{1}{2}^{N-1}$ unit vertical location.

8. A method according to claim 1 in which values for sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations are interpolated by taking the average of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

9. A method according to claim 1 in which values for sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations are interpolated by taking the average of values of a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

10. A method according to claim 1 in which values for half of the sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations are interpolated by taking the average of a first pair of values of a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location and values for the other half of the sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations are interpolated by taking the average of a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

11. A method according to claim 10 in which values for sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations are alternately interpolated for one such sub-pixel by taking the average of a first pair of values of a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location and values and for a neighbouring such sub-pixel by taking the average of a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

12. A method according to claim 11 in which the sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations are alternately interpolated in a horizontal direction.

13. A method according to claim 11 in which the sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations are alternately interpolated in a vertical direction.

14. A method according to claim 1 in which when values for some sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations are required, such values are interpolated by taking the average of a plurality of nearest neighbouring pixels.

15. A method according to claim 1 in which N equals an integer selected from a list consisting of the values 2, 3, and 4.

16. A method according to claim 1 in which in at least one of step a) and step b) interpolating sub-pixel values directly using weighted sums involves the calculation of an intermediate value for the sub-pixel values having a dynamic range greater than the specified dynamic range.

17. A method according to claim 14 in which the intermediate value for a sub-pixel having $\frac{1}{2}^{N-1}$ sub-pixel resolution is used the calculation of a sub-pixel value having $\frac{1}{2}^N$ sub-pixel resolution.

18. A method of interpolation in video coding in which an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, is interpolated to generate values for sub-pixels at fractional horizontal and vertical locations, the method comprising:

- a) when values for sub-pixels at half unit horizontal and unit vertical locations, and unit horizontal and half unit vertical locations are required, interpolating such values directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) when values for sub-pixels at half unit horizontal and half unit vertical locations are required, interpolating such values directly using a weighted sum of values for sub-pixels residing at half unit horizontal and unit vertical locations calculated according to step (a); and
- c) when values for sub-pixels at quarter unit horizontal and quarter unit vertical locations are required, interpolating such values by taking the average of at least one pair of a first pair of values of a sub-pixel located at a half unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and half unit vertical location and a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a half unit horizontal and half unit vertical location.

19. A method of interpolation in video coding in which an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, is interpolated to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $\frac{1}{2}^x$ where x is a positive integer having a maximum value N, the method comprising:

- a) when values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations are required, interpolating such values directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) when a value for a sub-pixel at a sub-pixel horizontal and sub-pixel vertical location is required, interpolating such a value directly using a choice of a first weighted sum of values for sub-pixels located at a vertical location corresponding to that of the sub-pixel being calculated and a second weighted sum of values for sub-pixels located at a horizontal location corresponding to that of the sub-pixel being calculated.

20. A method according to claim 1 in which the sub-pixels used in the first weighted sum are sub-pixels residing at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations and the first weighted sum is used to interpolate a value for a sub-pixel at $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^N$ unit vertical location.

21. A method according to claim 1 in which the sub-pixels used in the second weighted sum are sub-pixels residing at unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations and the second weighted sum is used to interpolate a value for a sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

22. A method according to claim 1 in which when values for sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical

locations are required, they are interpolated by taking the average of at least one pair of a first pair of values of a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location and a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

23. A video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the video coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $\frac{1}{2}^x$, where x is a positive integer having a maximum value N, the interpolator being adapted to:

- a) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using a choice of a first weighted sum of values for sub-pixels residing at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a); and
- c) interpolate a value for a sub-pixel situated at a $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical location by taking a weighted average of the value of a first sub-pixel or pixel situated at a $\frac{1}{2}^{N-m}$ unit horizontal and $\frac{1}{2}^{N-n}$ unit vertical location and the value of a second sub-pixel or pixel located at a $\frac{1}{2}^{N-p}$ unit horizontal and $\frac{1}{2}^{N-q}$ unit vertical location, variables m, n, p and q taking integer values in the range 1 to N such that the first and second sub-pixels or pixels are located diagonally with respect to the sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ vertical location.

24. A video coder according to claim 23, comprising a video encoder.

25. A video encoder according to claim 23, comprising a video decoder.

26. A codec comprising the video encoder of claim 24 or the video decoder of claim 25.

27. A communications terminal comprising a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the video coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $\frac{1}{2}^x$, where x is a positive integer having a maximum value N, the interpolator being adapted to:

- a) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using a choice

of a first weighted sum of values for sub-pixels residing at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a); and

- c) interpolate a value for a sub-pixel situated at a $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical location by taking a weighted average of the value of a first sub-pixel or pixel situated at a $\frac{1}{2}^{N-m}$ unit horizontal and $\frac{1}{2}^{N-n}$ unit vertical location and the value of a second sub-pixel or pixel located at a $\frac{1}{2}^{N-p}$ unit horizontal and $\frac{1}{2}^{N-q}$ unit vertical location, variables m, n, p and q taking integer values in the range 1 to N such that the first and second sub-pixels or pixels are located diagonally with respect to the sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ vertical location.

28. A communications terminal according to claim 27 comprising a video encoder.

29. A communications terminal according to claim 27 comprising a video decoder.

30. A communications terminal according to claim 27 having a video codec comprising a video encoder and a video decoder.

31. A telecommunications system comprising a communications terminal and a network, the telecommunications network and the communications terminal being connected by a communications link over which coded video can be transmitted, the communications terminal comprising a video coder for coding for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the video coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $\frac{1}{2}^x$, where x is a positive integer having a maximum value N, the interpolator being adapted to:

- a) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using a choice of a first weighted sum of values for sub-pixels residing at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a); and
- c) interpolate a value for a sub-pixel situated at a $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical location by taking a weighted average of the value of a first sub-pixel or pixel situated at a $\frac{1}{2}^{N-m}$ unit horizontal and $\frac{1}{2}^{N-n}$ unit vertical location and the value of a second sub-pixel or pixel located at a $\frac{1}{2}^{N-p}$ unit horizontal and $\frac{1}{2}^{N-q}$ unit vertical location, variables m, n, p and q taking integer values in the range 1 to N such that the first and second sub-pixels or pixels are located diagonally with respect to the sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ vertical location.

32. A telecommunications system according to claim 31 which is a mobile telecommunications system comprising a mobile communications terminal and a wireless network, the connection between the mobile communications terminal and the wireless network being formed by a radio link.

33. A telecommunications system according to claim 31 in which the network enables the communications terminal to communicate with other communications terminals connected to the network over communications links between the other communications terminals and the network.

34. A telecommunications system comprising a communications terminal and a network, the telecommunications network and the communications terminal being connected by a communications link over which coded video can be transmitted, the network comprising a video coder for coding for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the video coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $\frac{1}{2}^x$, where x is a positive integer having a maximum value N, the interpolator being adapted to:

- a) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using a choice of a first weighted sum of values for sub-pixels residing at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a); and
- c) interpolate a value for a sub-pixel situated at a $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical location by taking a weighted average of the value of a first sub-pixel or pixel situated at a $\frac{1}{2}^{N-m}$ unit horizontal and $\frac{1}{2}^{N-n}$ unit vertical location and the value of a second sub-pixel or pixel located at a $\frac{1}{2}^{N-p}$ unit horizontal and $\frac{1}{2}^{N-q}$ unit vertical location, variables m, n, p and q taking integer values in the range 1 to N such that the first and second sub-pixels or pixels are located diagonally with respect to the sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ vertical location.

35. A video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the resolution of the sub-pixels being determined by a positive integer N, the interpolator being adapted to:

- a) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate a value for a sub-pixel at a sub-pixel horizontal and sub-pixel vertical location is required directly using a choice of a first weighted sum of values for sub-pixels located at a vertical location corresponding to that of the sub-pixel being calculated and a second weighted sum of values for sub-pixels located at a horizontal location corresponding to that of the sub-pixel being calculated.

51

36. A video coder according to claim 35 in which the interpolator is further adapted to form the first weighted sum using the values of sub-pixels residing at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations and to use the first weighted sum to interpolate a value for a sub-pixel at $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^N$ unit vertical location.

37. A video coder according to claim 35 in which the interpolator is further adapted to form the second weighted sum using the values of sub-pixels residing at unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations and to use the second weighted sum to interpolate a value for a sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

38. A video coder according to claim 35 in which the interpolator is further adapted to interpolate values for sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations by taking the average of at least one pair of a first pair of values of a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location and a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

39. A video coder according to claim 35, comprising a video encoder.

40. A video decoder according to claim 35, comprising a video decoder.

41. A codec comprising the video encoder of claim 39 or the video decoder of claim 40.

42. A communications terminal comprising a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the resolution of the sub-pixels being determined by a positive integer N, the interpolator being adapted to:

- a) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate a value for a sub-pixel at a sub-pixel horizontal and sub-pixel vertical location is required directly using a choice of a first weighted sum of values for sub-pixels located at a vertical location corresponding to that of the sub-pixel being calculated and a second weighted sum of values for sub-pixels located at a horizontal location corresponding to that of the sub-pixel being calculated.

43. A communications terminal according to claim 42 comprising a video encoder.

44. A communications terminal according to claim 42 comprising a video decoder.

45. A communications terminal according to claim 42 having a video codec comprising a video encoder and a video decoder.

46. A telecommunications system comprising a communications terminal and a network, the telecommunications network and the communications terminal being connected by a communications link over which coded video can be transmitted, the communications terminal comprising a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns

52

residing at unit vertical locations, the coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the resolution of the sub-pixels being determined by a positive integer N, the interpolator being adapted to:

- a) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate a value for a sub-pixel at a sub-pixel horizontal and sub-pixel vertical location is required directly using a choice of a first weighted sum of values for sub-pixels located at a vertical location corresponding to that of the sub-pixel being calculated and a second weighted sum of values for sub-pixels located at a horizontal location corresponding to that of the sub-pixel being calculated.

47. A telecommunications system according to claim 46 which is a mobile telecommunications system comprising a mobile communications terminal and a wireless network, the connection between the mobile communications terminal and the wireless network being formed by a radio link.

48. A telecommunications system according to claim 46 in which the network enables the communications terminal to communicate with other communications terminals connected to the network over communications links between the other communications terminals and the network.

49. A telecommunications system comprising a communications terminal and a network, the telecommunications network and the communications terminal being connected by a communications link over which coded video can be transmitted, the network comprising a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the resolution of the sub-pixels being determined by a positive integer N, the interpolator being adapted to:

- a) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate a value for a sub-pixel at a sub-pixel horizontal and sub-pixel vertical location is required directly using a choice of a first weighted sum of values for sub-pixels located at a vertical location corresponding to that of the sub-pixel being calculated and a second weighted sum of values for sub-pixels located at a horizontal location corresponding to that of the sub-pixel being calculated.

50. A codec for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the codec comprising a video encoder and a video decoder, each of the video encoder and the video decoder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $\frac{1}{2}^x$, where x is a positive integer having a maximum value N, the interpolator of the video encoder and the interpolator of the video decoder each being adapted to:

53

- a) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
 - b) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using a choice of a first weighted sum of values for sub-pixels residing at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a); and
 - c) interpolate a value for a sub-pixel situated at a $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical location by taking a weighted average of the value of a first sub-pixel or pixel situated at a $\frac{1}{2}^{N-m}$ unit horizontal and $\frac{1}{2}^{N-n}$ unit vertical location and the value of a second sub-pixel or pixel located at a $\frac{1}{2}^{N-p}$ unit horizontal and $\frac{1}{2}^{N-q}$ unit vertical location, variables m, n, p and q taking integer values in the range 1 to N such that the first and second sub-pixels or pixels are located diagonally with respect to the sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ vertical location.
51. A codec for coding an image comprising pixels arranged in rows and columns and represented by values

54

having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the codec comprising a video encoder and a video decoder, each of the video encoder and the video decoder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the resolution of the sub-pixels being determined by a positive integer N, the interpolator of the video encoder and the interpolator of the video decoder each being adapted to:

- a) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate a value for a sub-pixel at a sub-pixel horizontal and sub-pixel vertical location directly using a choice of a first weighted sum of values for sub-pixels located at a vertical location corresponding to that of the sub-pixel being calculated and a second weighted sum of values for sub-pixels located at a horizontal location corresponding to that of the sub-pixel being calculated.

* * * * *

EXHIBIT 8

U.S. PATENT DOCUMENTS

5,452,377 A	9/1995	Igarashi	382/238
5,488,419 A	1/1996	Hui et al.	348/402
5,521,642 A	5/1996	Park	348/409
5,568,597 A	10/1996	Nakayama et al.	395/132
5,570,436 A	10/1996	Fukushima et al.	382/300
5,844,616 A	12/1998	Collet et al.	348/441
5,901,248 A	5/1999	Fandrianto et al.	382/236
6,219,464 B1	4/2001	Greggain et al.	382/298
6,252,576 B1	6/2001	Nottingham	345/127
6,950,469 B2 *	9/2005	Karczewicz et al. ...	375/240.17
2002/0064229 A1	5/2002	Nakaya	375/240.17

FOREIGN PATENT DOCUMENTS

GB 2205707 12/1988

WO

WO99/04574

1/1999

OTHER PUBLICATIONS

"Directional Interpolation for Magnetic Resonance Angiography Data", Mehran Mashfeghi, IEEE Transactions on Medical Imaging, vol. 12, No. 2, Jun. 1993.

Document VCEG-L20, "Complexity Reduced Motion Compensated Prediction with 1/8-pel Displacement Vector Resolution", ITU, Study Group 16.

English translation of Japanese publication No. Hei 6-14739.

Japanese Patent Document No. JP 62-039920.

* cited by examiner

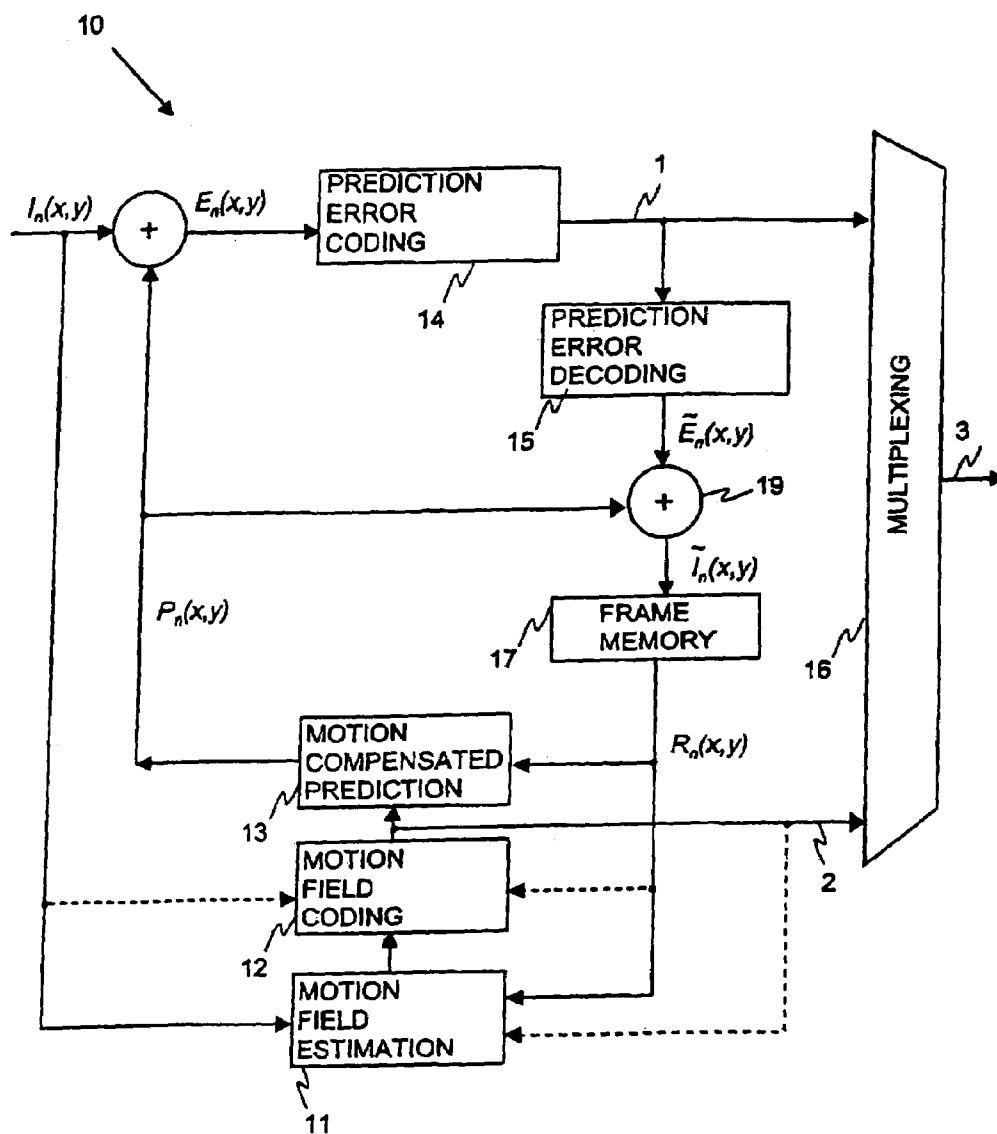
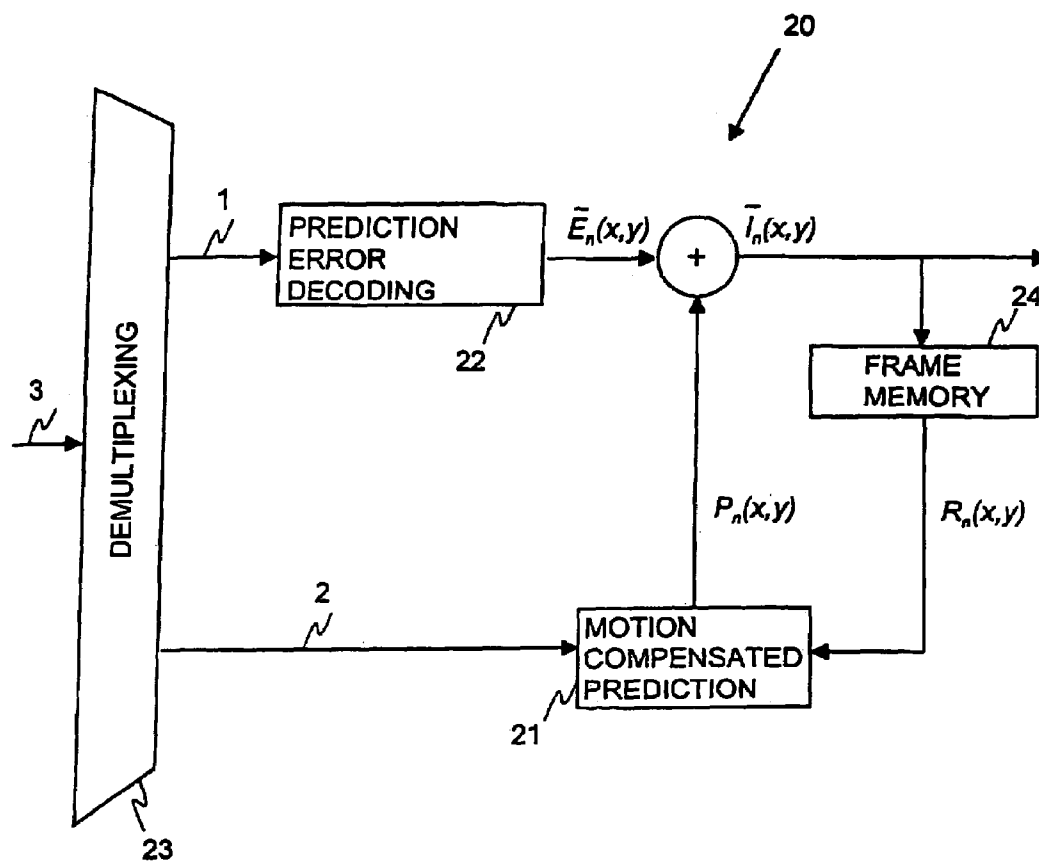


Fig. 1 PRIOR ART



PRIOR ART

Fig. 2

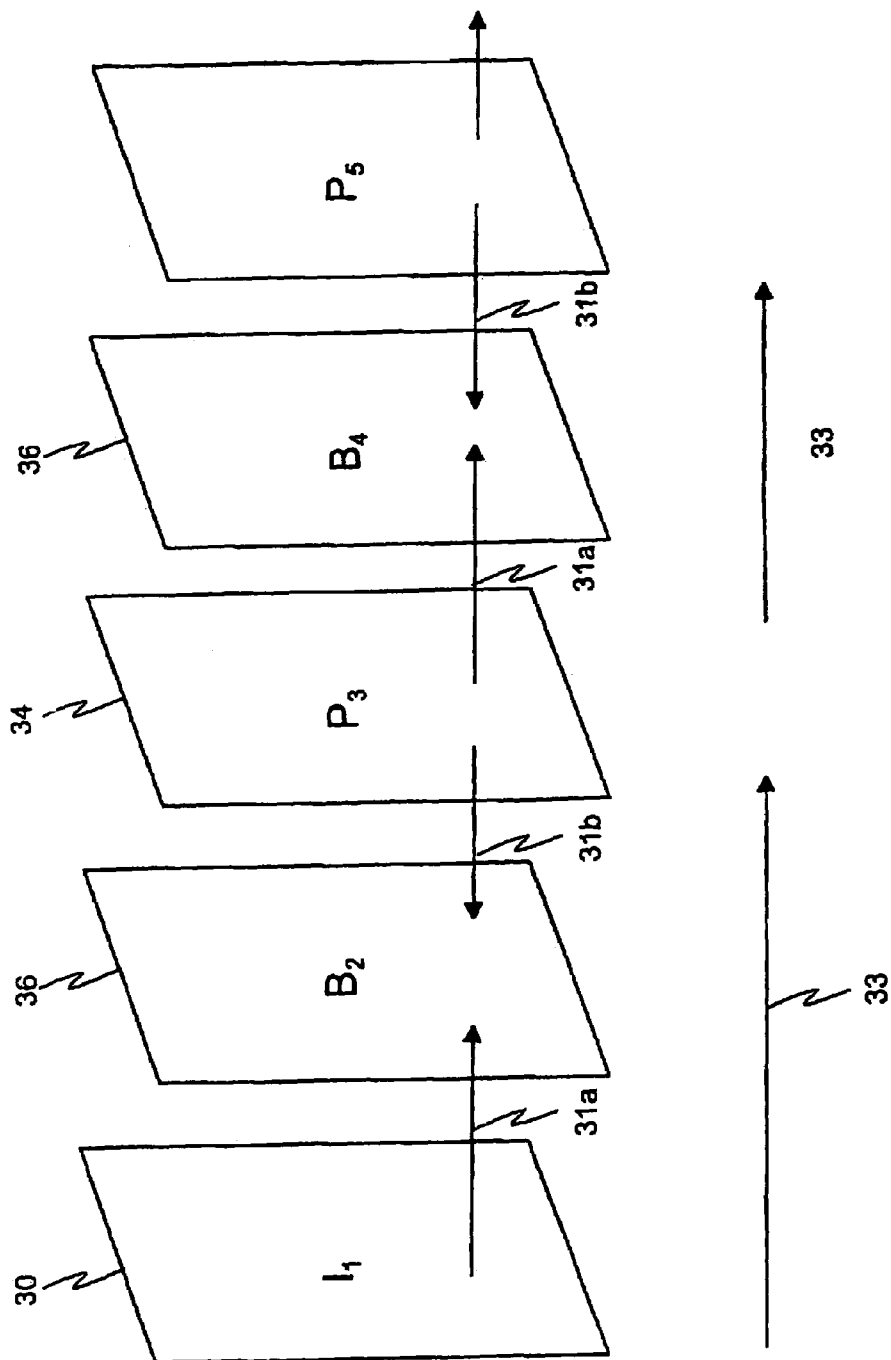


Fig. 3

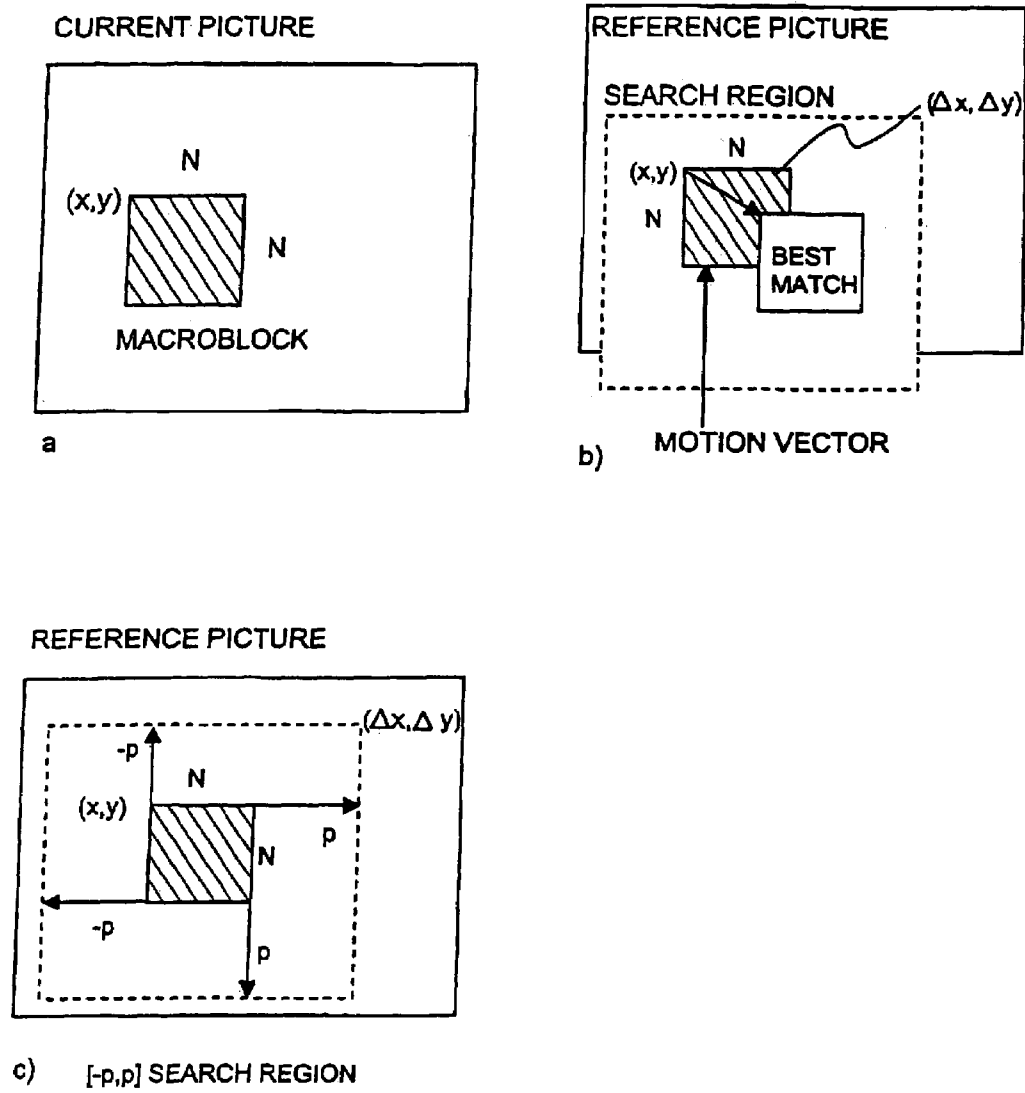


Fig. 4

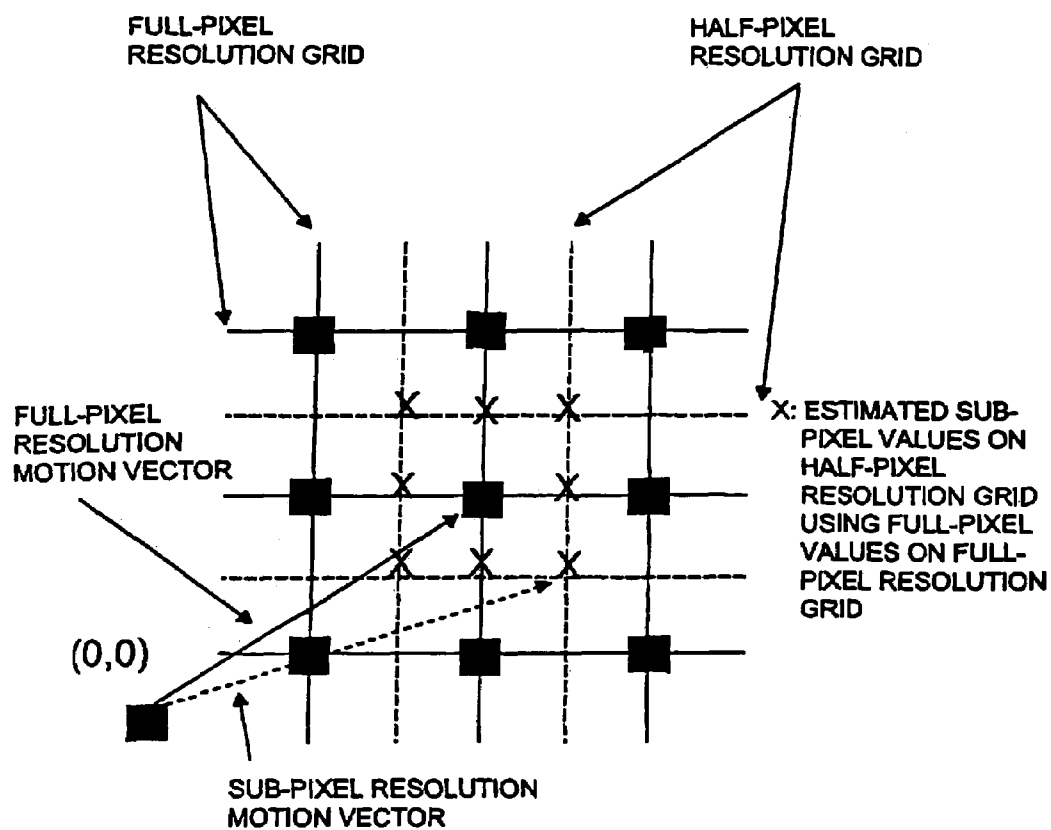


Fig. 5

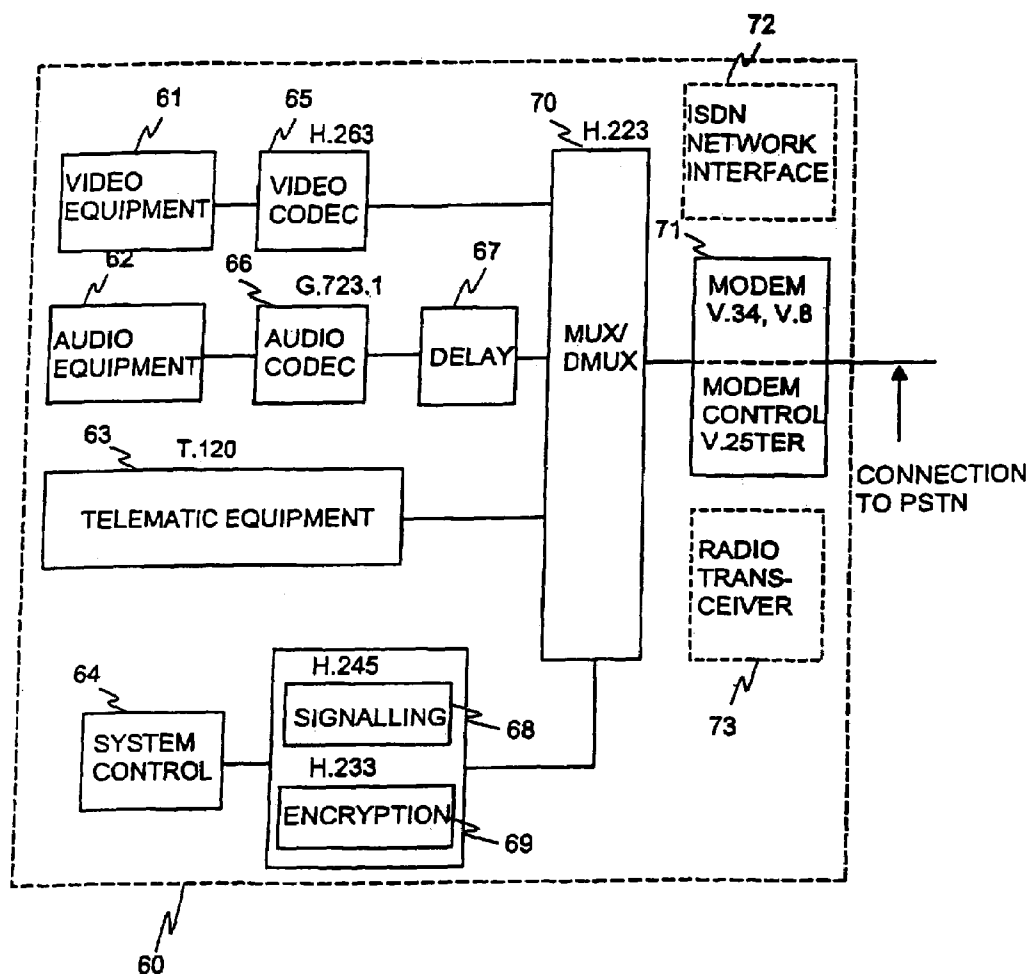


Fig. 6

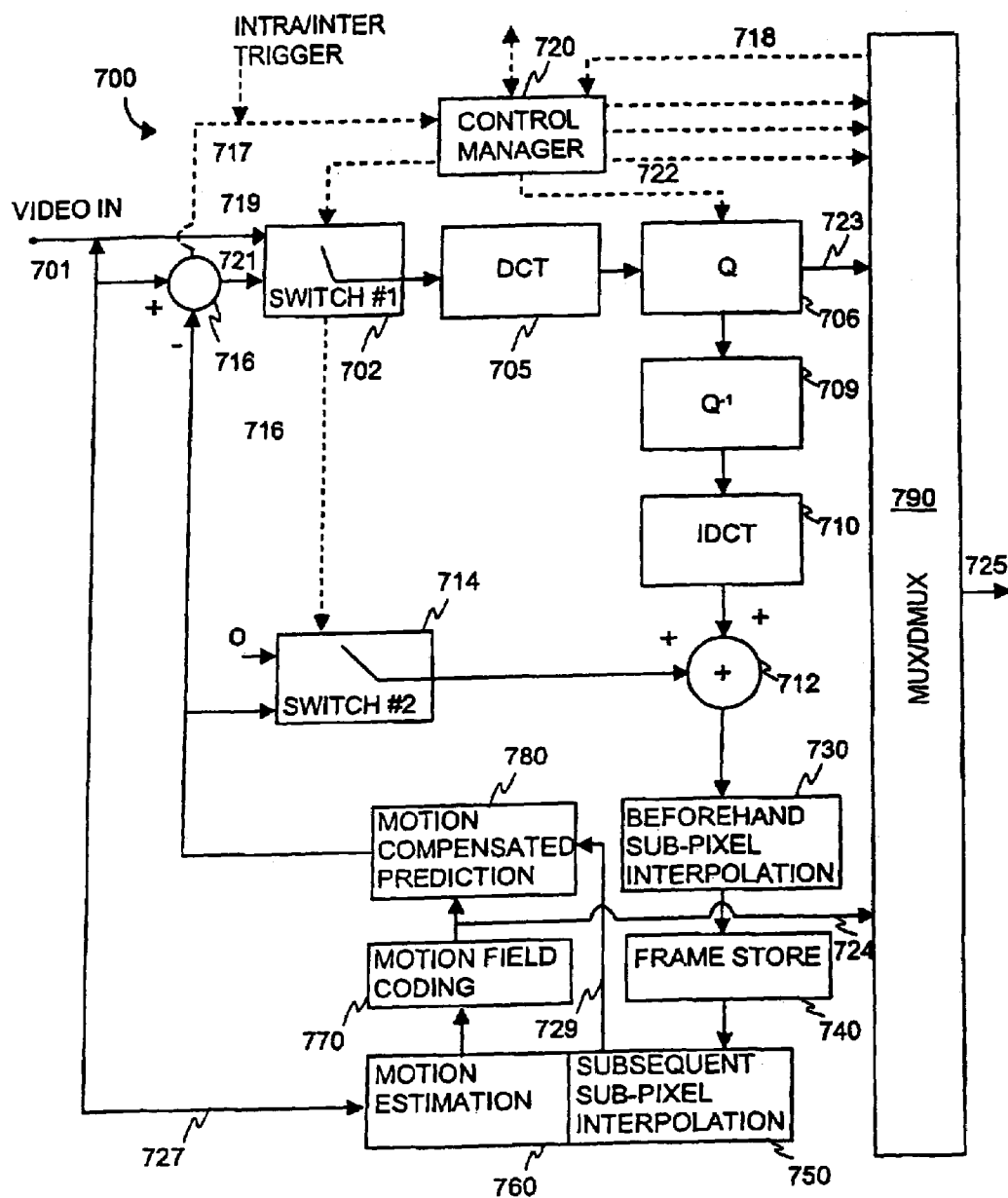


Fig. 7

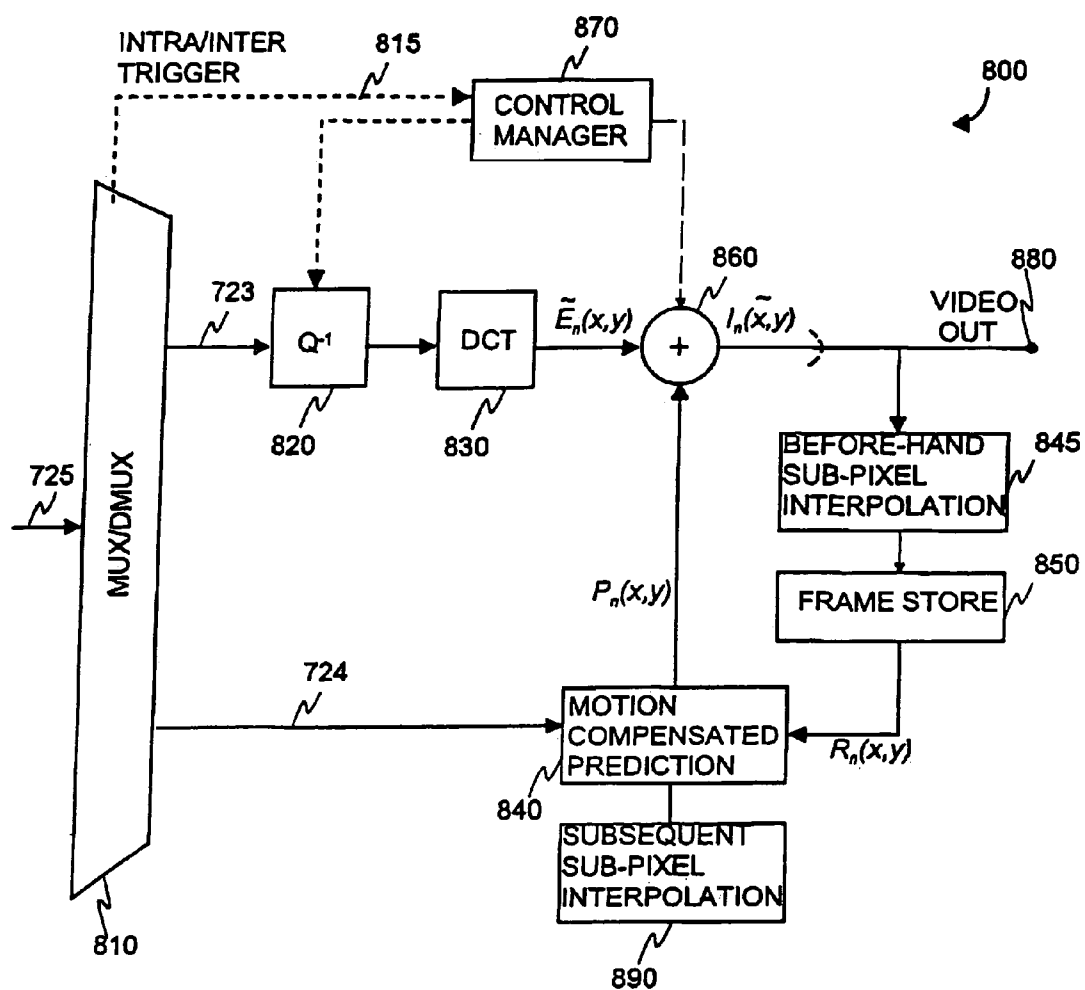


Fig. 8

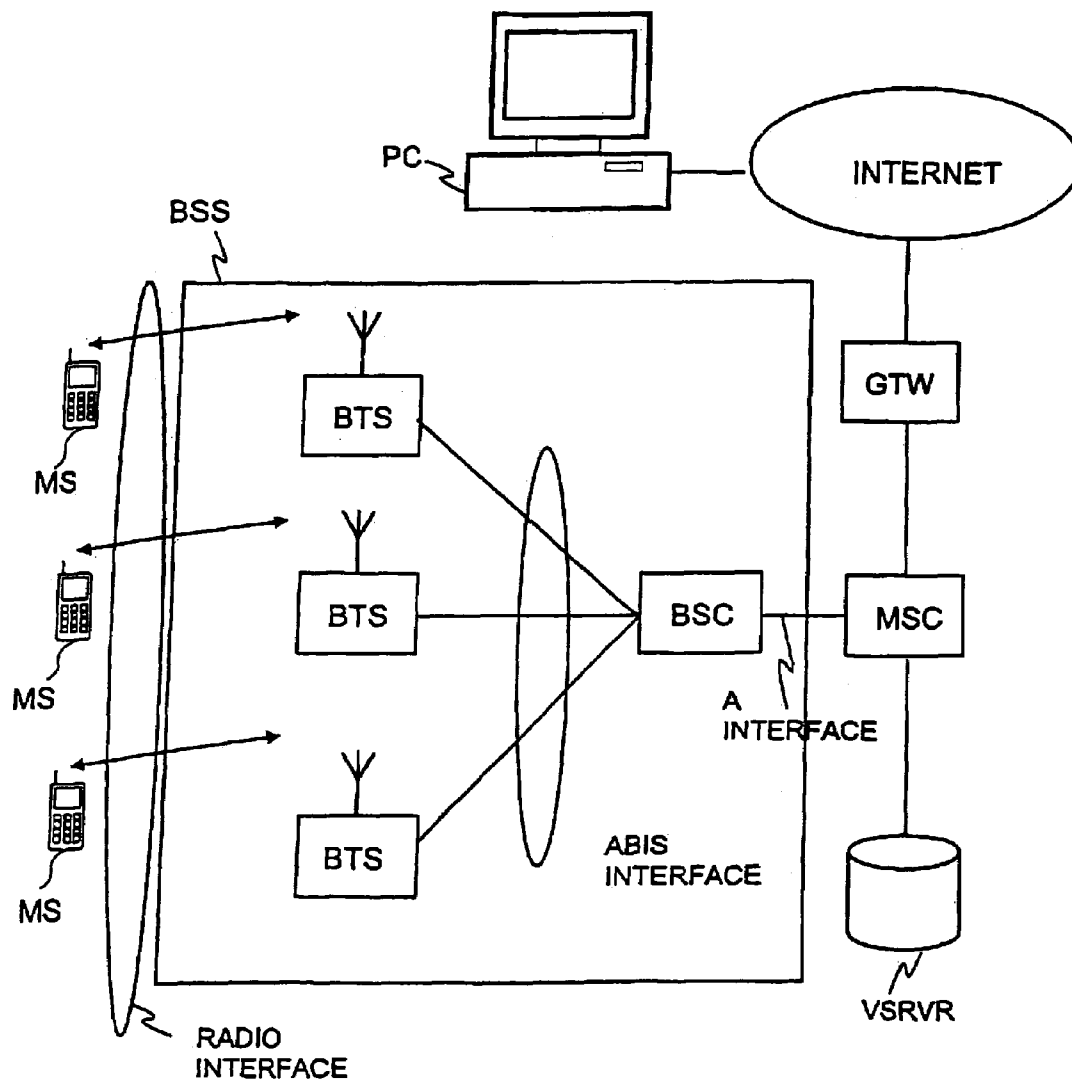


Fig. 11

A	d	b	d	A
e	e	e	e	
c	d	c	d	
e	e	e	f	
A				A

Fig. 12(a)

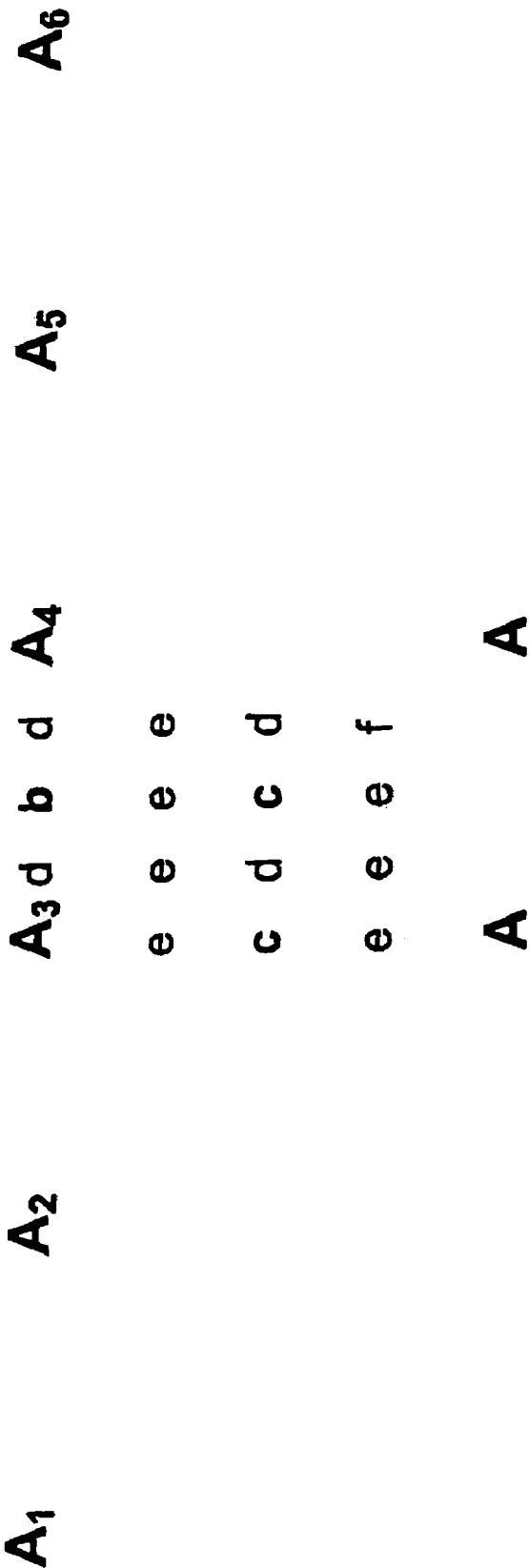


Figure 12(b)

A₁ b₁

A₂ b₂

A₃	d	b₃	d	A
e	e	e	e	
c	d	c	d	
e	e	e	f	
A₄		b₄		A

A₅ b₅

A₆ b₆

Figure 12(c)

A	d	b	d	A
e	g	e	g	
c	d	c	d	
e	g	e	f	
A				A

Fig. 13(a)

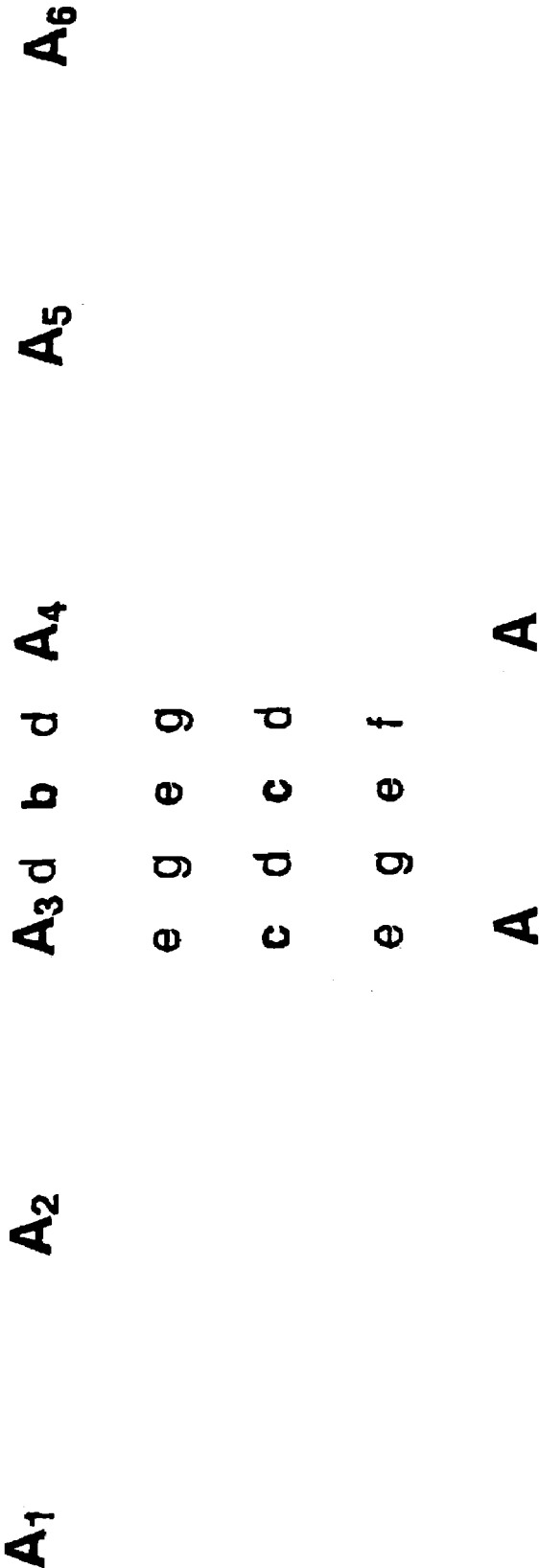


Figure13(b)

A₁ b₁

A₂ b₂

A₃	d	b₃	d	A
e	g	e	g	
c	d	c	d	
e	g	e	f	
A₄		b₄		A

A₅ b₅

A₆ b₆

Figure 13(c)

A	d	b	d	A
e	h	f	h	
b	g	c	g	
e	h	f	i	
A				A

Fig. 14(a)

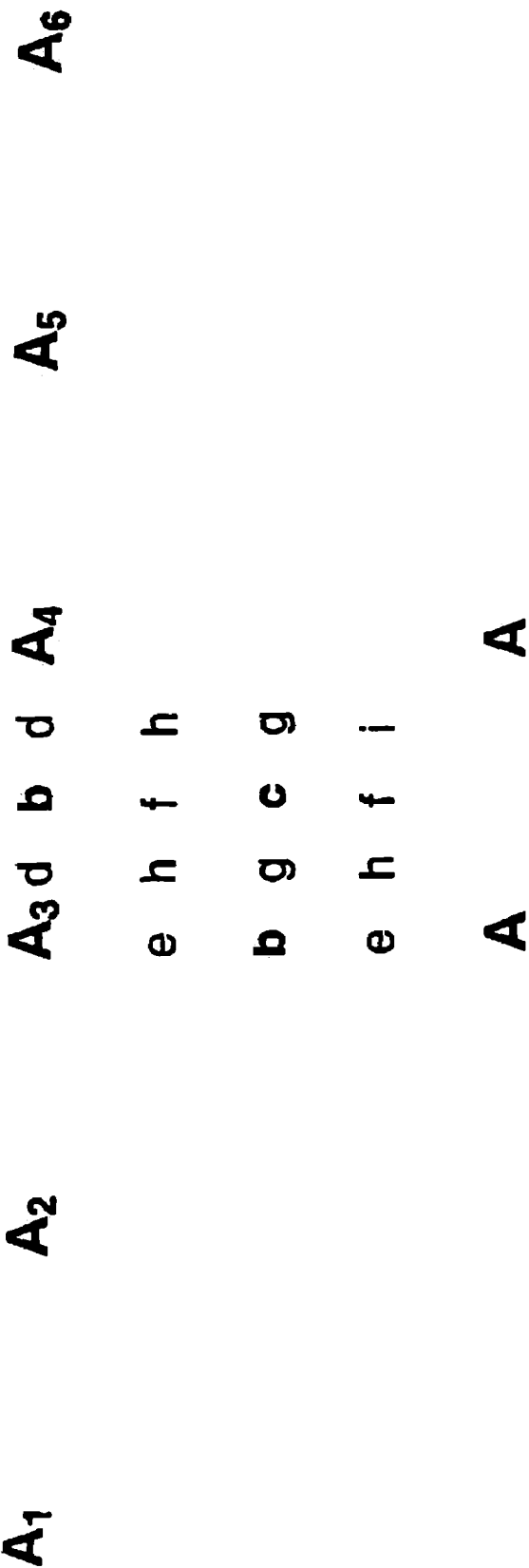


Figure 14(b)

A₁

A₂

A₃	d	b	d	A
e	h	f	h	
b	g	c	g	
e	h	f	i	
A₄				A

A₅

A₆

Figure 14(c)

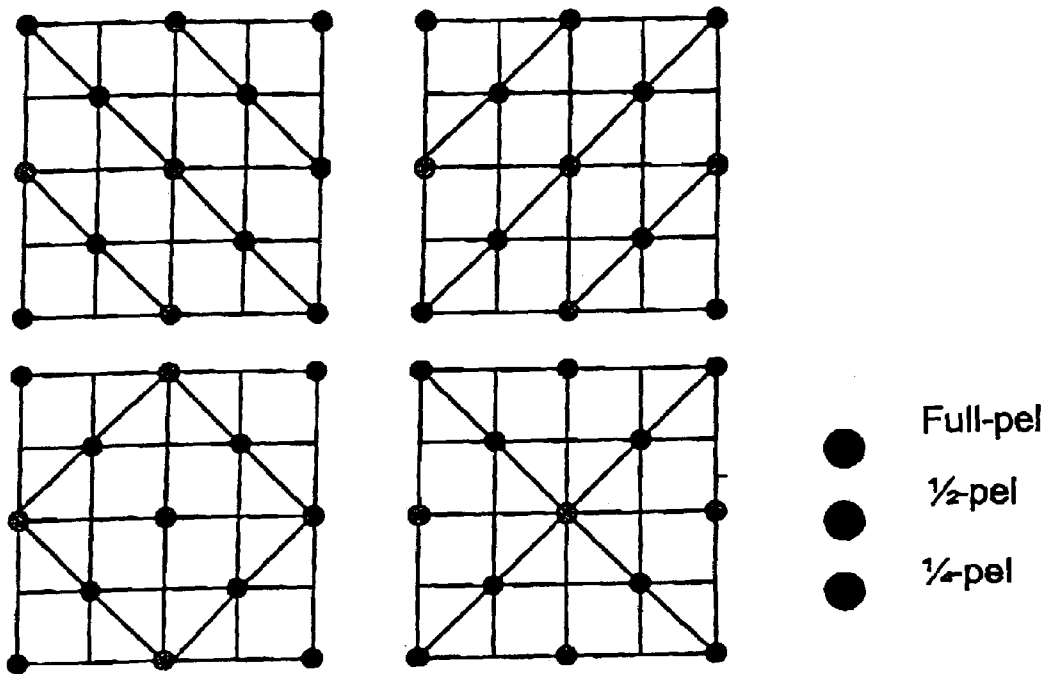


Fig. 15

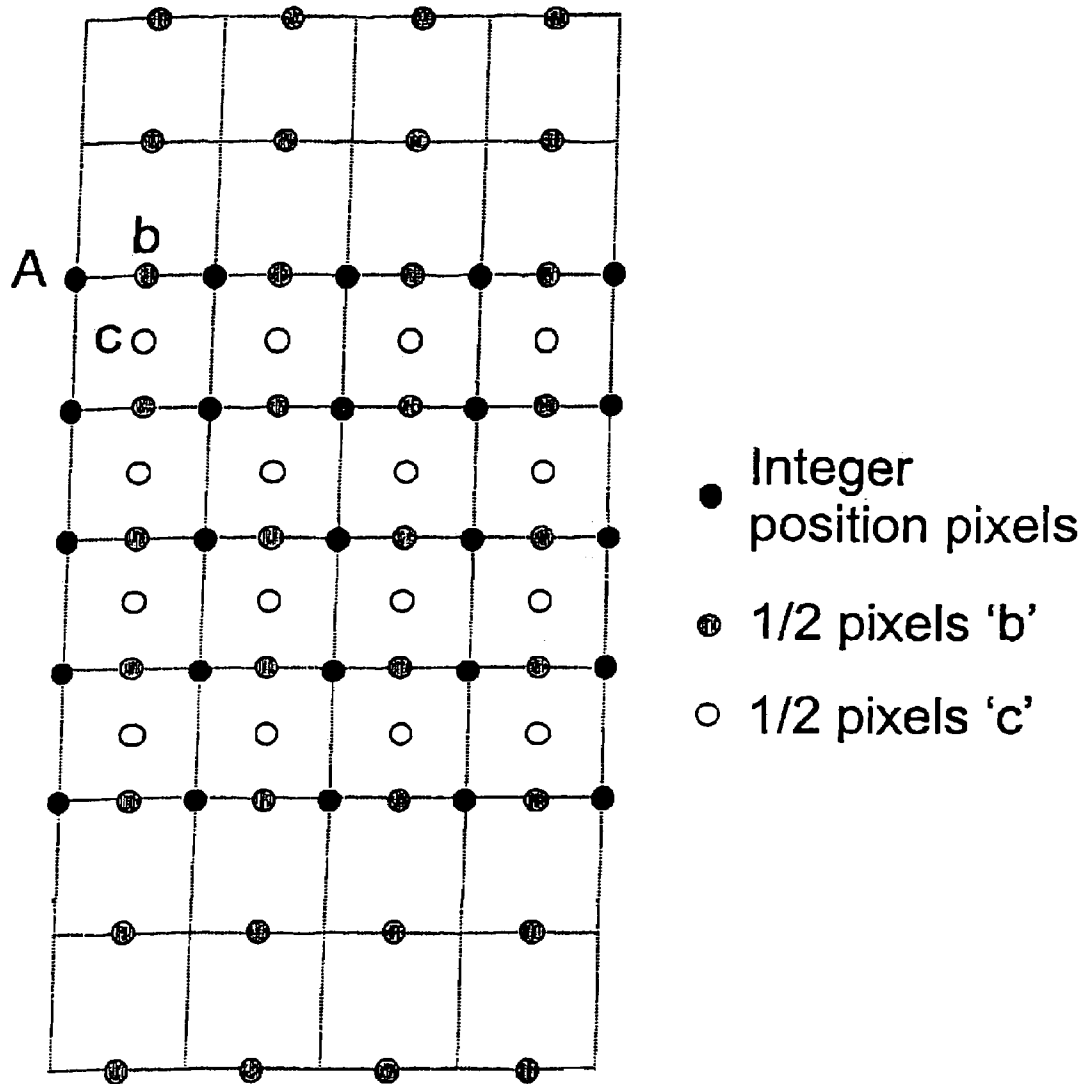


Fig. 16

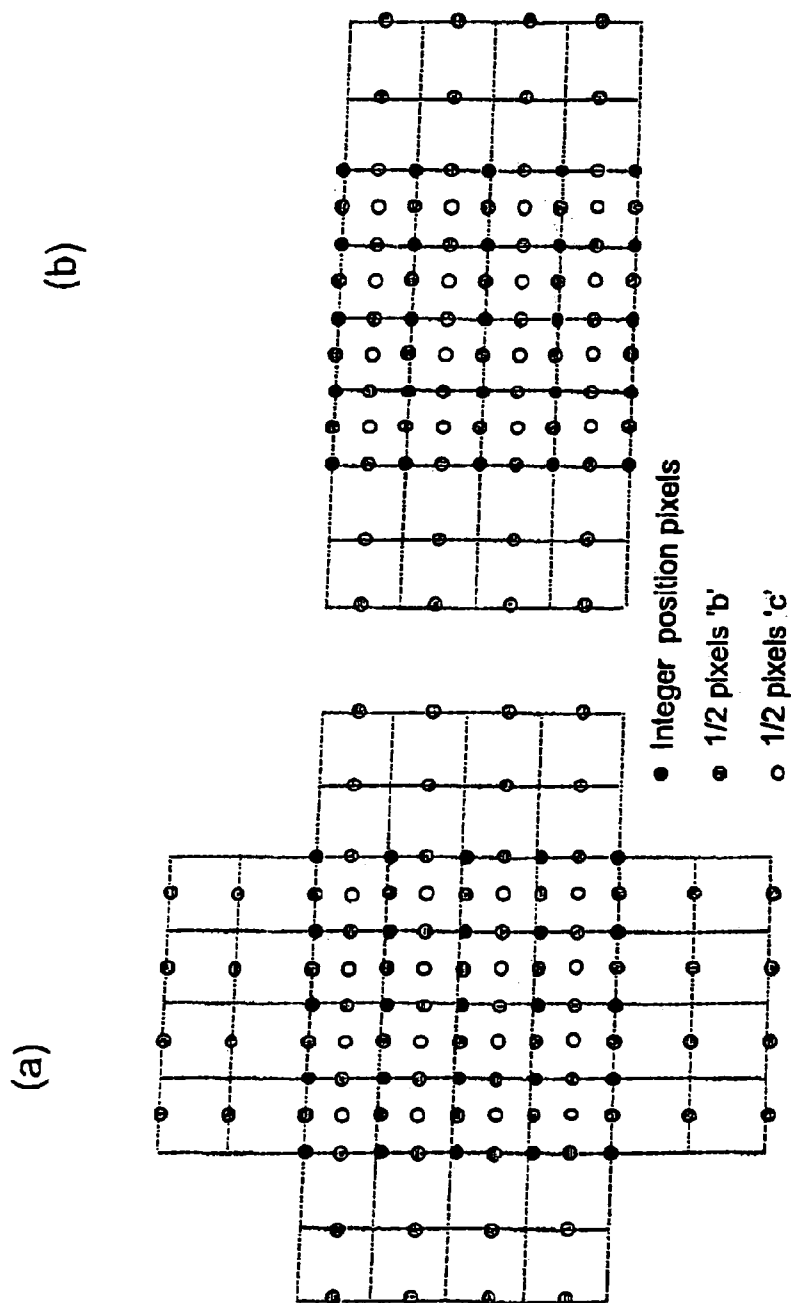


Fig. 17

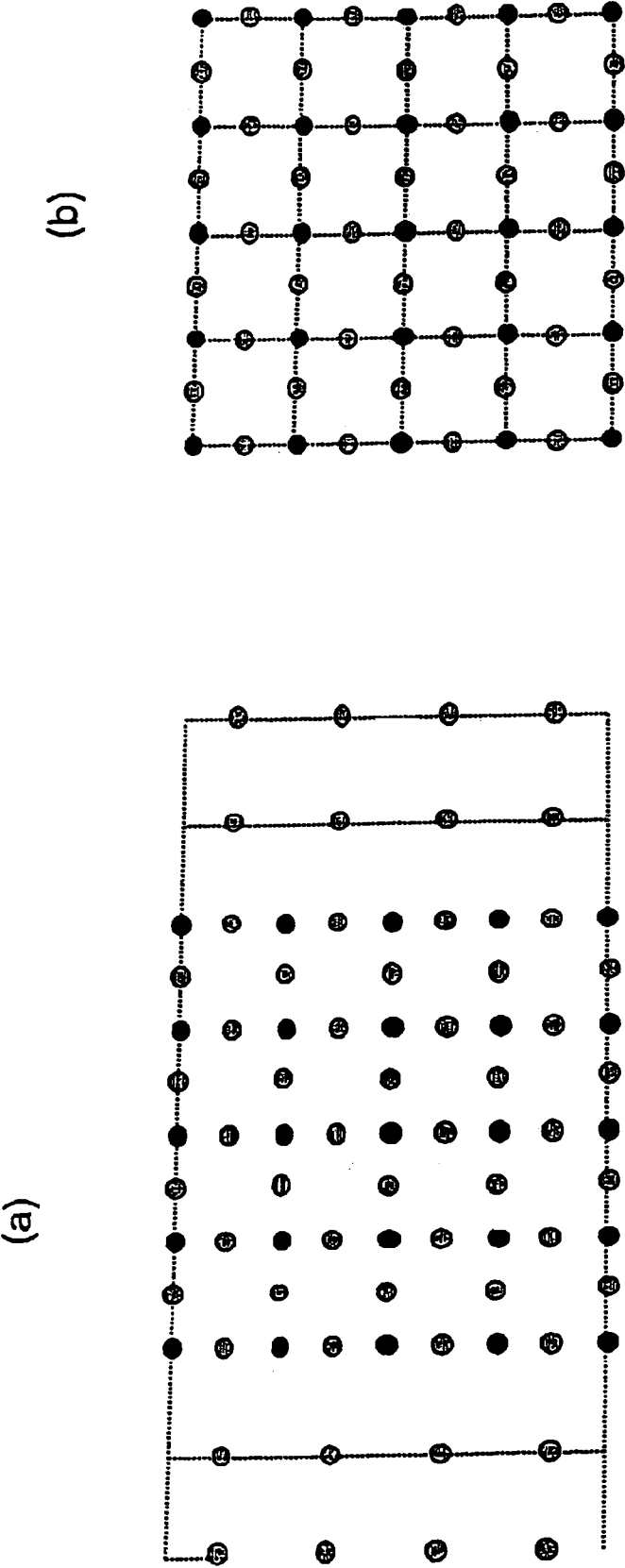


Fig. 18

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Fig. 19

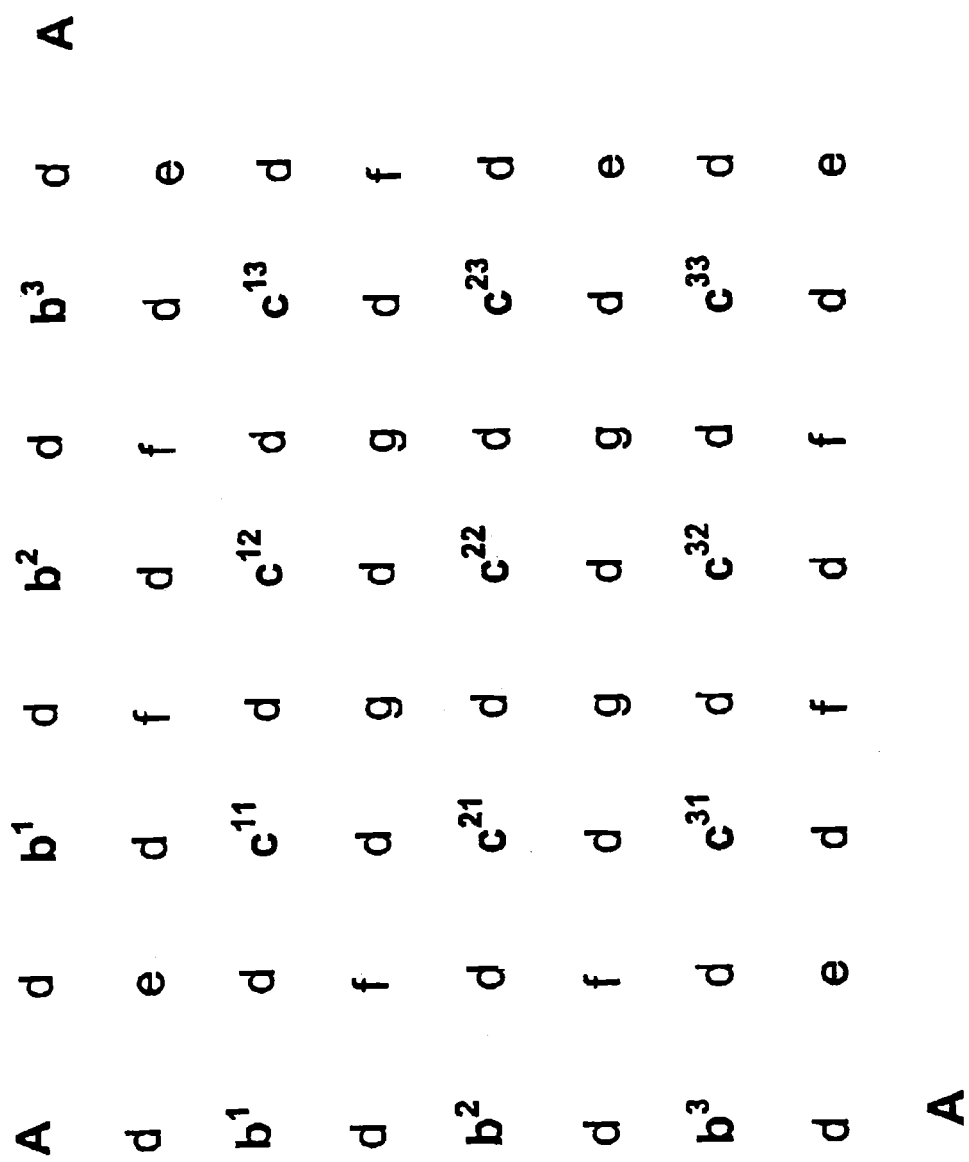


figure 20

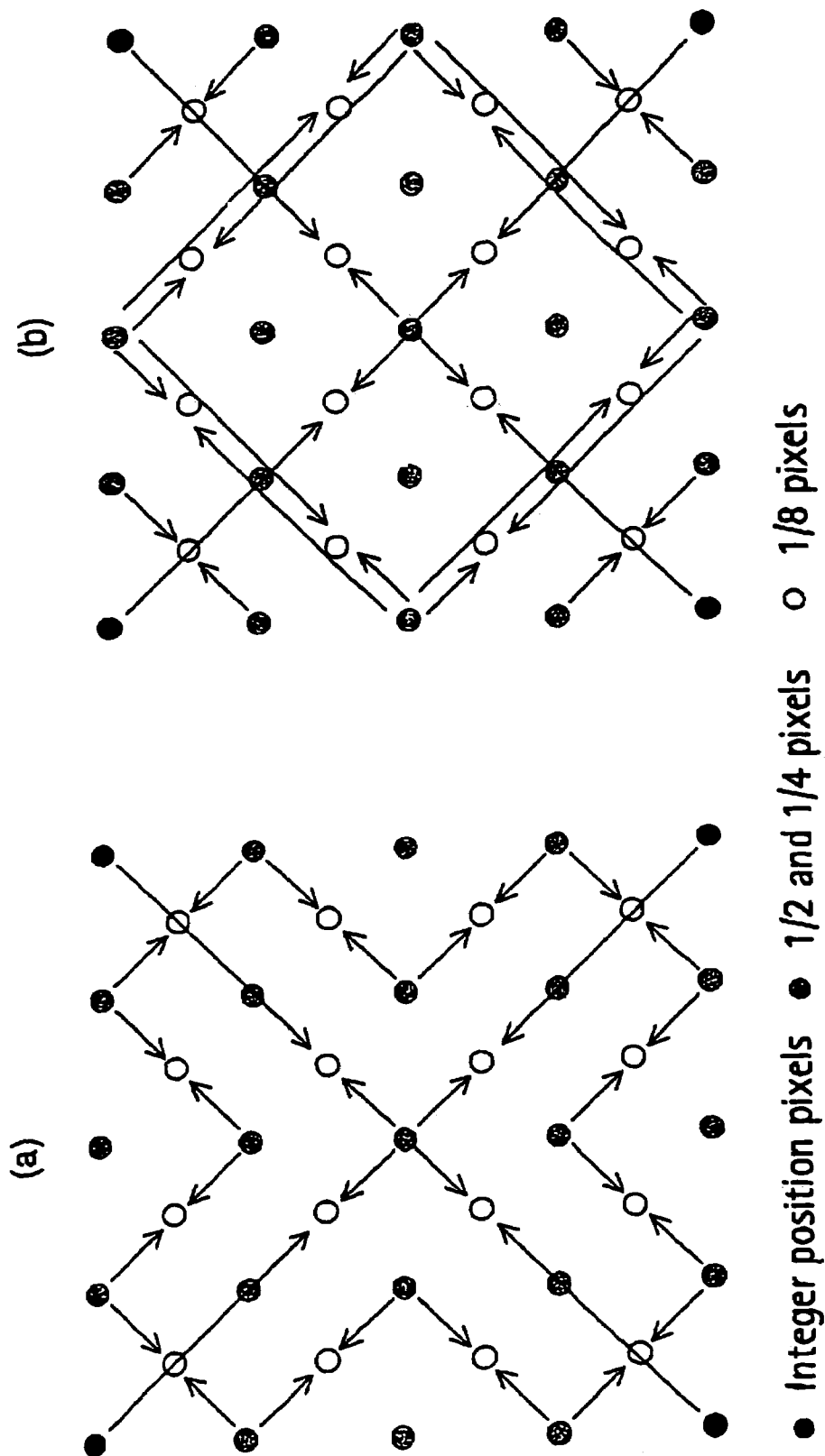


Fig. 21

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

Fig. 22

1

METHOD FOR SUB-PIXEL VALUE INTERPOLATION

CROSS REFERENCE TO RELATED APPLICATIONS

This application is a continuation of, and claims priority to, U.S. application Ser. No. 09/954,608, filed on Sep. 17, 2001, now U.S. Pat. No. 6,950,469.

The present invention relates to a method for sub-pixel value interpolation in the encoding and decoding of data. It relates particularly, but not exclusively, to encoding and decoding of digital video.

BACKGROUND OF THE INVENTION

Digital video sequences, like ordinary motion pictures recorded on film, comprise a sequence of still images, the illusion of motion being created by displaying the images one after the other at a relatively fast frame rate, typically 15 to 30 frames per second. Because of the relatively fast frame rate, images in consecutive frames tend to be quite similar and thus contain a considerable amount of redundant information. For example, a typical scene may comprise some stationary elements, such as background scenery, and some moving areas, which may take many different forms, for example the face of a newsreader, moving traffic and so on. Alternatively, the camera recording the scene may itself be moving, in which case all elements of the image have the same kind of motion. In many cases, this means that the overall change between one video frame and the next is rather small. Of course, this depends on the nature of the movement. For example, the faster the movement, the greater the change from one frame to the next. Similarly, if a scene contains a number of moving elements, the change from one frame to the next is likely to be greater than in a scene where only one element is moving.

It should be appreciated that each frame of a raw, that is uncompressed, digital video sequence comprises a very large amount of image information. Each frame of an uncompressed digital video sequence is formed from an array of image pixels. For example, in a commonly used digital video format, known as the Quarter Common Interchange Format (QCIF), a frame comprises an array of 176×144 pixels, in which case each frame has 25,344 pixels. In turn, each pixel is represented by a certain number of bits, which carry information about the luminance and/or colour content of the region of the image corresponding to the pixel. Commonly, a so-called YUV colour model is used to represent the luminance and chrominance content of the image. The luminance, or Y, component represents the intensity (brightness) of the image, while the colour content of the image is represented by two chrominance components, labelled U and V.

Colour models based on a luminance/chrominance representation of image content provide certain advantages compared with colour models that are based on a representation involving primary colours (that is Red, Green and Blue, RGB). The human visual system is more sensitive to intensity variations than it is to colour variations; YUV colour models exploit this property by using a lower spatial resolution for the chrominance components (U, V) than for the luminance component (Y). In this way the amount of information needed to code the colour information in an image can be reduced with an acceptable reduction in image quality.

2

The lower spatial resolution of the chrominance components is usually attained by sub-sampling. Typically, a block of 16×16 image pixels is represented by one block of 16×16 pixels comprising luminance information and the corresponding chrominance components are each represented by one block of 8×8 pixels representing an area of the image equivalent to that of the 16×16 pixels of the luminance component. The chrominance components are thus spatially sub-sampled by a factor of 2 in the x and y directions. The resulting assembly of one 16×16 pixel luminance block and two 8×8 pixel chrominance blocks is commonly referred to as a YUV macroblock, or macroblock, for short.

A QCIF image comprises 11×9 macroblocks. If the luminance blocks and chrominance blocks are represented with 8 bit resolution (that is by numbers in the range 0 to 255), the total number of bits required per macroblock is $(16 \times 16 \times 8) + 2 \times (8 \times 8 \times 8) = 3072$ bits. The number of bits needed to represent a video frame in QCIF format is thus $99 \times 3072 = 304,128$ bits. This means that the amount of data required to transmit record/display a video sequence in QCIF format, represented using a YUV colour model, at a rate of 30 frames per second, is more than 9 Mbps (million bits per second). This is an extremely high data rate and is impractical for use in video recording, transmission and display applications because of the very large storage capacity, transmission channel capacity and hardware performance required.

If video data is to be transmitted in real-time over a fixed line network such as an ISDN (Integrated Services Digital Network) or a conventional PSTN (Public Service Telephone Network), the available data transmission bandwidth is typically of the order of 64 kbits/s. In mobile videotelephony, where transmission takes place at least in part over a radio communications link, the available bandwidth can be as low as 20 kbits/s. This means that a significant reduction in the amount of information used to represent video data must be achieved in order to enable transmission of digital video sequences over low bandwidth communication networks. For this reason video compression techniques have been developed which reduce the amount of information transmitted while retaining an acceptable image quality.

Video compression methods are based on reducing the redundant and perceptually irrelevant parts of video sequences. The redundancy in video sequences can be categorised into spatial, temporal and spectral redundancy. 'Spatial redundancy' is the term used to describe the correlation between neighbouring pixels within a frame. The term 'temporal redundancy' expresses the fact that the objects appearing in one frame of a sequence are likely to appear in subsequent frames, while 'spectral redundancy' refers to the correlation between different colour components of the same image.

Sufficiently efficient compression cannot usually be achieved by simply reducing the various forms of redundancy in a given sequence of images. Thus, most current video encoders also reduce the quality of those parts of the video sequence which are subjectively the least important. In addition, the redundancy of the compressed video bit-stream is itself reduced by means of efficient loss-less encoding. Typically, this is achieved using a technique known as 'variable length coding' (VLC).

Modern video compression standards, such as ITU-T recommendations H.261, H.263(+)(++), H.26L and the Motion Picture Experts Group recommendation MPEG-4 make use of 'motion compensated temporal prediction'. This is a form of temporal redundancy reduction in which the content of some (often many) frames in a video sequence

is 'predicted' from other frames in the sequence by tracing the motion of objects or regions of an image between frames.

Compressed images which do not make use of temporal redundancy reduction are usually called INTRA-coded or I-frames, whereas temporally predicted images are called INTER-coded or P-frames. In the case of INTER frames, the predicted (motion-compensated) image is rarely precise enough to represent the image content with sufficient quality, and therefore a spatially compressed prediction error (PE) frame is also associated with each INTER frame. Many video compression schemes can also make use of bi-directionally predicted frames, which are commonly referred to as B-pictures or B-frames. B-pictures are inserted between reference or so-called 'anchor' picture pairs (I or P frames) and are predicted from either one or both of the anchor pictures. B-pictures are not themselves used as anchor pictures, that is no other frames are predicted from them, and therefore, they can be discarded from the video sequence without causing deterioration in the quality of future pictures.

The different types of frame that occur in a typical compressed video sequence are illustrated in FIG. 3 of the accompanying drawings. As can be seen from the figure, the sequence starts with an INTRA or I frame 30. In FIG. 3, arrows 33 denote the 'forward' prediction process by which P-frames (labelled 34) are formed. The bidirectional prediction process by which B-frames (36) are formed is denoted by arrows 31a and 31b, respectively.

A schematic diagram of an example video coding system using motion compensated prediction is shown in FIGS. 1 and 2. FIG. 1 illustrates an encoder 10 employing motion compensation and FIG. 2 illustrates a corresponding decoder 20. The encoder 10 shown in FIG. 1 comprises a Motion Field Estimation block 11, a Motion Field Coding block 12, a Motion Compensated Prediction block 13, a Prediction Error Coding block 14, a Prediction Error Decoding block 15, a Multiplexing block 16, a Frame Memory 17, and an adder 19. The decoder 20 comprises a Motion Compensated Prediction block 21, a Prediction Error Decoding block 22, a Demultiplexing block 23 and a Frame Memory 24.

The operating principle of video coders using motion compensation is to minimise the amount of information in a prediction error frame $E_n(x,y)$, which is the difference between a current frame $I_n(x,y)$ being coded and a prediction frame $P_n(x,y)$. The prediction error frame is thus:

$$E_n(x,y) = I_n(x,y) - P_n(x,y). \quad (1)$$

The prediction frame $P_n(x,y)$ is built using pixel values of a reference frame $R_n(x,y)$, which is generally one of the previously coded and transmitted frames, for example the frame immediately preceding the current frame and is available from the Frame Memory 17 of the encoder 10. More specifically, the prediction frame $P_n(x,y)$ is constructed by finding so-called 'prediction pixels' in the reference frame $R_n(x,y)$ which correspond substantially with pixels in the current frame. Motion information, describing the relationship (e.g. relative location, rotation, scale etc.) between pixels in the current frame and their corresponding prediction pixels in the reference frame is derived and the prediction frame is constructed by moving the prediction pixels according to the motion information. In this way, the prediction frame is constructed as an approximate representation of the current frame, using pixel values in the reference frame. The prediction error frame referred to above therefore represents the difference between the approximate representation of the current frame provided by the prediction frame

and the current frame itself. The basic advantage provided by video encoders that use motion compensated prediction arises from the fact that a comparatively compact description of the current frame can be obtained by representing it in terms of the motion information required to form its prediction together with the associated prediction error information in the prediction error frame.

However, due to the very large number of pixels in a frame, it is generally not efficient to transmit separate motion information for each pixel to the decoder. Instead, in most video coding schemes, the current frame is divided into larger image segments S_k and motion information relating to the segments is transmitted to the decoder. For example, motion information is typically provided for each macroblock of a frame and the same motion information is then used for all pixels within the macroblock. In some video coding standards, such as H.26L, a macroblock can be divided into smaller blocks, each smaller block being provided with its own motion information.

The motion information usually takes the form of motion vectors $[\Delta x(x,y), \Delta y(x,y)]$. The pair of numbers $\Delta x(x,y)$ and $\Delta y(x,y)$ represents the horizontal and vertical displacements of a pixel at location (x,y) in the current frame $I_n(x,y)$ with respect to a pixel in the reference frame $R_n(x,y)$. The motion vectors $[\Delta x(x,y), \Delta y(x,y)]$ are calculated in the Motion Field Estimation block 11 and the set of motion vectors of the current frame $[\Delta x(\cdot), \Delta y(\cdot)]$ is referred to as the motion vector field.

Typically, the location of a macroblock in a current video frame is specified by the (x,y) co-ordinate of its upper left-hand corner. Thus, in a video coding scheme in which motion information is associated with each macroblock of a frame, each motion vector describes the horizontal and vertical displacement $\Delta x(x,y)$ and $\Delta y(x,y)$ of a pixel representing the upper left-hand corner of a macroblock in the current frame $I_n(x,y)$ with respect to a pixel in the upper left-hand corner of a substantially corresponding block of prediction pixels in the reference frame $R_n(x,y)$ (as shown in FIG. 4b).

Motion estimation is a computationally intensive task. Given a reference frame $R_n(x,y)$ and, for example, a square macroblock comprising $N \times N$ pixels in a current frame (as shown in FIG. 4a), the objective of motion estimation is to find an $N \times N$ pixel block in the reference frame that matches the characteristics of the macroblock in the current picture according to some criterion. This criterion can be, for example, a sum of absolute differences (SAD) between the pixels of the macroblock in the current frame and the block of pixels in the reference frame with which it is compared. This process is known generally as 'block matching'. It should be noted that, in general, the geometry of the block to be matched and that in the reference frame do not have to be the same, as real-world objects can undergo scale changes, as well as rotation and warping. However, in current international video coding standards, only a translational motion model is used (see below) and thus fixed rectangular geometry is sufficient.

Ideally, in order to achieve the best chance of finding a match, the whole of the reference frame should be searched. However, this is impractical as it imposes too high a computational burden on the video encoder. Instead, the search region is restricted to region $[-p, p]$ around the original location of the macroblock in the current frame, as shown in FIG. 4c.

In order to reduce the amount of motion information to be transmitted from the encoder 10 to the decoder 20, the motion vector field is coded in the Motion Field Coding

5

block 12 of the encoder 10, by representing it with a motion model. In this process, the motion vectors of image segments are re-expressed using certain predetermined functions or, in other words, the motion vector field is represented with a model. Almost all currently used motion vector field models are additive motion models, complying with the following general formula:

$$\Delta x(x, y) = \sum_{i=0}^{N-1} a_i f_i(x, y) \quad (2)$$

$$\Delta y(x, y) = \sum_{i=0}^{M-1} b_i g_i(x, y) \quad (3)$$

where coefficients a_i and b_i are called motion coefficients. The motion coefficients are transmitted to the decoder 20 (information stream 2 in FIGS. 1 and 2). Functions f_i and g_i are called motion field basis functions, and are known both to the encoder and decoder. An approximate motion vector field ($\Delta x(x, y), \Delta y(x, y)$) can be constructed using the coefficients and the basis functions. As the basis functions are known to (that is stored in) both the encoder 10 and the decoder 20, only the motion coefficients need to be transmitted to the encoder, thus reducing the amount of information required to represent the motion information of the frame.

The simplest motion model is the translational motion model which requires only two coefficients to describe the motion vectors of each segment. The values of motion vectors are given by:

$$\begin{aligned} \Delta x(x, y) &= a_0 \\ \Delta y(x, y) &= b_0 \end{aligned} \quad (4)$$

This model is widely used in various international standards (ISO MPEG-1, MPEG-2, MPEG4, ITU-T Recommendations H.261 and H.263) to describe the motion of 16x16 and 8x8 pixel blocks. Systems which use a translational motion model typically perform motion estimation at full pixel resolution or some integer fraction of full pixel resolution, for example at half or one quarter pixel resolution.

The prediction frame $P_n(x, y)$ is constructed in the Motion Compensated Prediction block 13 in the encoder 10, and is given by:

$$P_n(x, y) = R_n[x + \Delta x(x, y), y + \Delta y(x, y)] \quad (5)$$

In the Prediction Error Coding block 14, the prediction error frame $E_n(x, y)$ is typically compressed by representing it as a finite series (transform) of some 2-dimensional functions. For example, a 2-dimensional Discrete Cosine Transform (DCT) can be used. The transform coefficients are quantised and entropy (for example Huffman) coded before they are transmitted to the decoder (information stream 1 in FIGS. 1 and 2). Because of the error introduced by quantisation, this operation usually produces some degradation (loss of information) in the prediction error frame $E_n(x, y)$. To compensate for this degradation, the encoder 10 also comprises a Prediction Error Decoding block 15, where a decoded prediction error frame $\tilde{E}_n(x, y)$ is constructed using the transform coefficients. This locally decoded prediction error frame is added to the prediction frame $P_n(x, y)$ in the adder 19 and the resulting decoded current frame $\tilde{I}_n(x, y)$ is stored in the Frame Memory 17 for further use as the next reference frame $R_{n+1}(x, y)$.

6

The information stream 2 carrying information about the motion vectors is combined with information about the prediction error in multiplexer 16 and an information stream 3 containing typically at least those two types of information is sent to the decoder 20.

The operation of a corresponding video decoder 20 will now be described.

The Frame Memory 24 of the decoder 20 stores a previously reconstructed reference frame $R_n(x, y)$. The prediction frame $P_n(x, y)$ is constructed in the Motion Compensated Prediction block 21 of the decoder 20 according to equation 5, using received motion coefficient information and pixel values of the previously reconstructed reference frame $R_n(x, y)$. The transmitted transform coefficients of the prediction error frame $E_n(x, y)$ are used in the Prediction Error Decoding block 22 to construct the decoded prediction error frame $\tilde{E}_n(x, y)$. The pixels of the decoded current frame $\tilde{I}_n(x, y)$ are then reconstructed by adding the prediction frame $P_n(x, y)$ and the decoded prediction error frame $\tilde{E}_n(x, y)$:

$$\tilde{I}_n(x, y) = P_n(x, y) + \tilde{E}_n(x, y) = R_n[x + \Delta x(x, y), y + \Delta y(x, y)] + \tilde{E}_n(x, y). \quad (6)$$

This decoded current frame may be stored in the Frame Memory 24 as the next reference frame $R_{n+1}(x, y)$.

In the description of motion compensated encoding and decoding of digital video presented above, the motion vector $[\Delta x(x, y), \Delta y(x, y)]$ describing the motion of a macroblock in the current frame with respect to the reference frame $R_n(x, y)$ can point to any of the pixels in the reference frame. This means that motion between frames of a digital video sequence can only be represented at a resolution which is determined by the image pixels in the frame (so-called full pixel resolution). Real motion, however, has arbitrary precision, and thus the system described above can only provide approximate modelling of the motion between successive frames of a digital video sequence. Typically, modelling of motion between video frames with full pixel resolution is not sufficiently accurate to allow efficient minimisation of the prediction error (PE) information associated with each macroblock/frame. Therefore, to enable more accurate modelling of real motion and to help reduce the amount of PE information that must be transmitted from encoder to decoder, many video coding standards, such as H.263(+) and H.26L, allow motion vectors to point 'in between' image pixels. In other words, the motion vectors can have 'sub-pixel' resolution. Allowing motion vectors to have sub-pixel resolution adds to the complexity of the encoding and decoding operations that must be performed, so it is still advantageous to limit the degree of spatial resolution a motion vector may have. Thus, video coding standards, such as those previously mentioned, typically only allow motion vectors to have full-, half- or quarter-pixel resolution.

Motion estimation with sub-pixel resolution is usually performed as a two-stage process, as illustrated in FIG. 5, for a video coding scheme which allows motion vectors to have full- or half-pixel resolution. In the first step, a motion vector having full-pixel resolution is determined using any appropriate motion estimation scheme, such as the block-matching process described in the foregoing. The resulting motion vector, having full-pixel resolution is shown in FIG. 5.

In the second stage, the motion vector determined in the first stage is refined to obtain the desired half-pixel resolution. In the example illustrated in FIG. 5, this is done by forming eight new search blocks of 16x16 pixels, the location of the top-left corner of each block being marked with an X in FIG. 5. These locations are denoted as $[\Delta x + m/2, \Delta y + n/2]$, where m and n can take the values -1, 0

and +1, but cannot be zero at the same time. As only the pixel values of original image pixels are known, the values (for example luminance and/or chrominance values) of the sub-pixels residing at half-pixel locations must be estimated for each of the eight new search blocks, using some form of interpolation scheme.

Having interpolated the values of the sub-pixels at half-pixel resolution, each of the eight search blocks is compared with the macroblock whose motion vector is being sought. As in the block matching process performed in order to determine the motion vector with full pixel resolution, the macroblock is compared with each of the eight search blocks according to some criterion, for example a SAD. As a result of the comparisons, a minimum SAD value will generally be obtained. Depending on the nature of the motion in the video sequence, this minimum value may correspond to the location specified by the original motion vector (having full-pixel resolution), or it may correspond to a location having a half-pixel resolution. Thus, it is possible to determine whether a motion vector should point to a full-pixel or sub-pixel location and if sub-pixel resolution is appropriate, to determine the correct sub-pixel resolution motion vector. It should also be appreciated that the scheme just described can be extended to other sub-pixel resolutions (for example, one-quarter-pixel resolution) in an entirely analogous fashion.

In practice, the estimation of a sub-pixel value in the reference frame is performed by interpolating the value of the sub-pixel from surrounding pixel values. In general, interpolation of a sub-pixel value $F(x,y)$ situated at a non-integer location $(x, y) = (n + \Delta x, m + \Delta y)$, can be formulated as a two-dimensional operation, represented mathematically as:

$$F(x, y) = \sum_{k=-K}^{K-1} \sum_{l=-L}^{L-1} f(k + K, l + L) F(n + k, m + l) \quad (7)$$

where $f(k,l)$ are filter coefficients and n and m are obtained by truncating x and y , respectively, to integer values. Typically, the filter coefficients are dependent on the x and y values and the interpolation filters are usually so-called 'separable filters', in which case sub-pixel value $F(x,y)$ can be calculated as follows:

$$F(x, y) = \sum_{k=-K}^{K-1} f(k + K) \sum_{l=-K}^{K-1} f(l + K) F(n + k, m + l) \quad (8)$$

The motion vectors are calculated in the encoder. Once the corresponding motion coefficients are transmitted to the decoder, it is a straightforward matter to interpolate the required sub-pixels using an interpolation method identical to that used in the encoder. In this way, a frame following a reference frame in the Frame Memory 24, can be reconstructed from the reference frame and the motion vectors.

The simplest way of applying sub-pixel value interpolation in a video coder is to interpolate each sub-pixel value every time it is needed. However, this is not an efficient solution in a video encoder, because it is likely that the same sub-pixel value will be required several times and thus calculations to interpolate the same sub-pixel value will be

performed multiple times. This results in an unnecessary increase of computational complexity/burden in the encoder.

An alternative approach, which limits the complexity of the encoder, is to pre-calculate and store all sub-pixel values in a memory associated with the encoder. This solution is called interpolation 'before-hand' interpolation hereafter in this document. While limiting complexity, before-hand interpolation has the disadvantage of increasing memory usage by a large margin. For example, if the motion vector accuracy is one quarter pixel in both horizontal and vertical dimensions, storing pre-calculated sub-pixel values for a complete image results in a memory usage that is 16 times that required to store the original, non-interpolated image. In addition, it involves the calculation of some sub-pixels which might not actually be required in calculating motion vectors in the encoder. Before-hand interpolation is also particularly inefficient in a video decoder, as the majority of pre-calculated sub-pixel values will never be required by the decoder. Thus, it is advantageous not to use pre-calculation in the decoder.

So-called 'on-demand' interpolation can be used to reduce memory requirements in the encoder. For example, if the desired pixel precision is quarter pixel resolution, only sub-pixels at one half unit resolution are interpolated before-hand for the whole frame and stored in the memory. Values of one-quarter pixel resolution sub-pixels are only calculated during the motion estimation/compensation process as and when it is required. In this case memory usage is only 4 times that required to store the original, non-interpolated image.

It should be noted that when before-hand interpolation is used, the interpolation process constitutes only a small fraction of the total encoder computational complexity/burden, since every pixel is interpolated just once. Therefore, in the encoder, the complexity of the interpolation process itself is not very critical when before-hand sub-pixel value interpolation is used. On the other hand, on-demand interpolation poses a much higher computational burden on the encoder, since sub-pixels may be interpolated many times. Hence the complexity of interpolation process, which may be considered in terms of the number of computational operations or operational cycles that must be performed in order to interpolate the sub-pixel values, becomes an important consideration.

In the decoder, the same sub-pixel values are used a few times at most and some are not needed at all. Therefore, in the decoder it is advantageous not to use before-hand interpolation at all, that is, it is advantageous not to pre-calculate any sub-pixel values.

Two interpolation schemes have been developed as part of the work ongoing in the ITU-Telecommunications Standardization Sector, Study Group 16, Video Coding Experts Group (VCEG), Questions 6 and 15. These approaches were proposed for incorporation into ITU-T recommendation H.26L and have been implemented in test models (TML) for the purposes of evaluation and further development. The test model corresponding to Question 15 is referred to as Test Model 5 (TML5), while that resulting from Question 6 is known as Test Model 6 (TML6). The interpolation schemes proposed in both TML5 and TML6 will now be described.

Throughout the description of the sub-pixel value interpolation scheme used in test model TML5, reference will be made to FIG. 12a, which defines a notation for describing pixel and sub-pixel locations specific to TML5. A separate notation, defined in FIG. 13a, will be used in the discussion of the sub-pixel value interpolation scheme used in TML6. A still further notation, illustrated in FIG. 14a, will be used

later in the text in connection with the sub-pixel value interpolation method according to the invention. It should be appreciated that the three different notations used in the text are intended to assist in the understanding of each interpolation method and to help distinguish differences between them. However, in all three figures, the letter A is used to denote original image pixels (full pixel resolution). More specifically, the letter A represents the location of pixels in the image data representing a frame of a video sequence, the pixel values of pixels A being either received as current frame $I_n(x,y)$ from a video source, or reconstructed and stored as a reference frame $R_n(x,y)$ in the Frame Memory 17, 24 of the encoder 10 or the decoder 20. All other letters represent sub-pixel locations, the values of the sub-pixels situated at the sub-pixel locations being obtained by interpolation.

Certain other terms will also be used in a consistent manner throughout the text to identify particular pixel and sub-pixel locations. These are as follows:

The term 'unit horizontal location' is used to describe the location of any sub-pixel that is constructed in a column of the original image data. Sub-pixels c and e in FIGS. 12a and 13a, as well as sub-pixels b and e in FIG. 14a have unit horizontal locations.

The term 'unit vertical location' is used to describe any sub-pixel that is constructed in a row of the original image data. Sub-pixels b and d in FIGS. 12a and 13a as well as sub-pixels b and d in FIG. 14a have unit vertical locations.

By definition, pixels A have unit horizontal and unit vertical locations.

The term 'half horizontal location' is used to describe the location of any sub-pixel that is constructed in a column that lies at half pixel resolution. Sub-pixels b, c, and e shown in FIGS. 12a and 13a fall into this category, as do sub-pixels b, c and f in FIG. 14a. In a similar manner, the term 'half vertical location' is used to describe the location of any sub-pixel that is constructed in a row that lies at half-pixel resolution, such as sub-pixels c and d in FIGS. 12a and 13a, as well as sub-pixels b, c and g in FIG. 14a.

Furthermore, the term 'quarter horizontal location' refers to any sub-pixel that is constructed in a column which lies at quarter-pixel resolution, such as sub-pixels d and e in FIG. 12a, sub-pixels d and g in FIG. 13a and sub-pixels d, g and h in FIG. 14a. Analogously, the term 'quarter vertical location' refers to sub-pixels that are constructed in a row which lies at quarter-pixel resolution. In FIG. 12a, sub-pixels e and f fall into this category, as do sub-pixels e, f and g in FIG. 13a and sub-pixels e, f and h in FIG. 14a.

The definition of each of the terms described above is shown by 'envelopes' drawn on the corresponding figures.

It should further be noted that it is often convenient to denote a particular pixel with a two-dimensional reference. In this case, the appropriate two-dimensional reference can be obtained by examining the intersection of the envelopes in FIGS. 12a, 13a and 14a. Applying this principle, pixel d in FIG. 12a, for example, has a half horizontal and half vertical location and sub-pixel e has a unit horizontal and quarter vertical location. In addition, and for ease of reference, sub-pixels that reside at half unit horizontal and unit vertical locations, unit horizontal and half unit vertical locations as well as half unit horizontal and half unit vertical locations, will be referred to as $\frac{1}{2}$ resolution sub-pixels. Sub-pixels which reside at any quarter unit horizontal and/or quarter unit vertical location will be referred to as $\frac{1}{4}$ resolution sub-pixels.

It should also be noted that in the descriptions of the two test models and in the detailed description of the invention

itself, it will be assumed that pixels have a minimum value of 0 and a maximum value of 2^n-1 where n is the number of bits reserved for a pixel value. The number of bits is typically 8. After a sub-pixel has been interpolated, if the value of that interpolated sub-pixel exceeds the value of 2^n-1 , it is restricted to the range of $[0, 2^n-1]$, i.e. values lower than the minimum allowed value will become the minimum value (0) and values larger than the maximum will become maximum value (2^n-1). This operation is called clipping.

The sub-pixel value interpolation scheme according to TML5 will now be described in detail with reference to FIGS. 12a, 12b and 12c.

1. The value for the sub-pixel at half unit horizontal and unit vertical location, that is $\frac{1}{2}$ resolution sub-pixel b in FIG. 12a, is calculated using a 6-tap filter. The filter interpolates a value for $\frac{1}{2}$ resolution sub-pixel b based upon the values of the 6 pixels (A_1 to A_6) situated in a row at unit horizontal locations and unit vertical locations symmetrically about b, as shown in FIG. 12b, according to the formula $b=(A_1-5A_2+20A_3+20A_4-5A_5+A_6+16)/32$. The operator/denotes division with truncation. The result is clipped to lie in the range $[0, 2^n-1]$.
2. Values for the $\frac{1}{2}$ resolution sub-pixels labelled c are calculated using the same six tap filter as used in step 1 and the six nearest pixels or sub-pixels (A or b) in the vertical direction. Referring now to FIG. 12c, the filter interpolates a value for the $\frac{1}{2}$ resolution sub-pixel c located at unit horizontal and half vertical location based upon the values of the 6 pixels (A_1 to A_6) situated in a column at unit horizontal locations and unit vertical locations symmetrically about c, according to the formula $c=(A_1-5A_2+20A_3+20A_4-5A_5+A_6+16)/32$. Similarly, a value for the $\frac{1}{2}$ resolution sub-pixel c at half horizontal and half vertical location is calculated according to $c=(b_1-5b_2+20b_3+20b_4-5b_5+b_6+16)/32$. Again, the operator/denotes division with truncation. The values calculated for the c sub-pixels are further clipped to lie in the range $[0, 2^n-1]$.

At this point in the interpolation process the values of all $\frac{1}{2}$ resolution sub-pixels have been calculated and the process proceeds to the calculation of $\frac{1}{4}$ resolution sub-pixel values.

3. Values for the $\frac{1}{4}$ resolution sub-pixels labelled d are calculated using linear interpolation and the values of the nearest pixels and/or $\frac{1}{2}$ resolution sub-pixels in the horizontal direction. More specifically, values for $\frac{1}{4}$ resolution sub-pixels d located at quarter horizontal and unit vertical locations, are calculated by taking the average of the immediately neighbouring pixel at unit horizontal and unit vertical location (pixel A) and the immediately neighbouring $\frac{1}{2}$ resolution sub-pixel at half horizontal and unit vertical location (sub-pixel b), i.e. according to $d=(A+b)/2$. Values for $\frac{1}{4}$ resolution sub-pixels d located at quarter horizontal and half vertical locations, are calculated by taking the average of the immediately neighbouring $\frac{1}{2}$ resolution sub-pixels c which lie at unit horizontal and half vertical location and half horizontal and half vertical locations respectively, i.e. according to $d=(c_1+c_2)/2$. Again operator/indicates division with truncation.
4. Values for the $\frac{1}{4}$ resolution sub-pixels labelled e are calculated using linear interpolation and the values of the nearest pixels and/or $\frac{1}{2}$ resolution sub-pixels in the vertical direction. In particular, $\frac{1}{4}$ resolution sub-pixels e at unit horizontal and quarter vertical locations are

11

calculated by taking the average of the immediately neighbouring pixel at unit horizontal and unit vertical location (pixel A) and the immediately neighbouring sub-pixel at unit horizontal and half vertical location (sub-pixel c) according to $e=(A+c)/2$. $1/4$ resolution sub-pixels e_3 at half horizontal and quarter vertical locations are calculated by taking the average of the immediately neighbouring sub-pixel at half horizontal and unit vertical location (sub-pixel b) and the immediately neighbouring sub-pixel at half horizontal and half vertical location (sub-pixel c), according to $e=(b+c)/2$. Furthermore, $1/4$ resolution sub-pixels e at quarter horizontal and quarter vertical locations are calculated by taking the average of the immediately neighbouring sub-pixels at quarter horizontal and unit vertical location and the corresponding sub-pixel at quarter horizontal and half vertical location (sub-pixels d), according to $e=(d_1+d_2)/2$. Once more, operator/indicates division with truncation.

- 5 The value for $1/4$ resolution sub-pixel f is interpolated by averaging the values of the 4 closest pixels values at unit horizontal and vertical locations, according to $f=(A_1+A_2+A_3+A_4+2)/4$, where pixels A_1 , A_2 , A_3 and A_4 are the four nearest original pixels.

A disadvantage of TML5 is that the decoder is computationally complex. This results from the fact that TML5 uses an approach in which interpolation of $1/4$ resolution sub-pixel values depends upon the interpolation of $1/2$ resolution sub-pixel values. This means that in order to interpolate the values of the $1/4$ resolution sub-pixels, the values of the $1/2$ resolution sub-pixels from which they are determined must be calculated first. Furthermore, since the values of some of the $1/4$ resolution sub-pixels depend upon the interpolated values obtained for other $1/4$ resolution sub-pixels, truncation of the $1/4$ resolution sub-pixel values has a deleterious effect on the precision of some of the $1/4$ resolution sub-pixel values. Specifically, the $1/4$ resolution sub-pixel values are less precise than they would be if calculated from values that had not been truncated and clipped. Another disadvantage of TML5 is that it is necessary to store the values of the $1/2$ resolution sub-pixels in order to interpolate the $1/4$ resolution sub-pixel values. Therefore, excess memory is required to store a result which is not ultimately required.

The sub-pixel value interpolation scheme according to TML6, referred to herein as direct interpolation, will now be described. In the encoder the interpolation method according to TML6 works like the previously described TML5 interpolation method, except that maximum precision is retained throughout. This is achieved by using intermediate values which are neither rounded nor clipped. A step-by-step description of interpolation method according to TML6 as applied in the encoder is given below with reference to FIGS. 13a, 13b and 13c.

1. The value for the sub-pixel at half unit horizontal and unit vertical location, that is $1/2$ resolution sub-pixel b in FIG. 13a, is obtained by first calculating an intermediate value b using a six tap filter. The filter calculates b based upon the values of the 6 pixels (A_1 to A_6) situated in a row at unit horizontal locations and unit vertical locations symmetrically about b, as shown in FIG. 13b, according to the formula $b=(A_1-5A_2+20A_3+20A_4-5A_5+A_6)$. The final value of b is then calculated as $b=(b+16)/32$ and is clipped to lie in the range $[0, 2^n-1]$. As before, the operator/denotes division with truncation.
2. Values for the $1/2$ resolution sub-pixels labelled c are obtained by first: calculating intermediate values c.

12

Referring to FIG. 13c, an intermediate value c for the $1/2$ resolution sub-pixel c located at unit horizontal and half vertical location is calculated based upon the values of the 6 pixels (A_1 to A_6) situated in a column at unit horizontal locations and unit vertical locations symmetrically about c, according to the formula $c=(A_1-5A_2+20A_3+20A_4-5A_5+A_6)$. The final value for the $1/2$ resolution sub-pixel c located at unit horizontal and half vertical location is then calculated according to $c=(c+16)/32$. Similarly, an intermediate value c for the $1/2$ resolution sub-pixel c at half horizontal and half vertical location is calculated according to $c=(b_1-5b_2+20b_3+20b_4-5b_5+b_6)$. A final value for this $1/2$ resolution sub-pixel is then calculated according to $(c+512)/1024$. Again, the operator/denotes division with truncation and the values calculated for $1/2$ resolution sub-pixels c are further clipped to lie in the range $[0, 2^n-1]$.

3. Values for the $1/4$ resolution sub-pixels labelled d are calculated as follows. Values for $1/4$ resolution sub-pixels d located at quarter horizontal and unit vertical locations, are calculated from the value of the immediately neighbouring pixel at unit horizontal and unit vertical location (pixel A) and the intermediate value b calculated in step (1) for the immediately neighbouring $1/2$ resolution sub-pixel at half horizontal and unit vertical location ($1/2$ resolution sub-pixel b), according to $d=(32A+b+32)/64$. Values for $1/4$ resolution sub-pixels d located at quarter horizontal and half vertical locations, are interpolated using the intermediate values c calculated for the immediately neighbouring $1/2$ resolution sub-pixels c which lie at unit horizontal and half vertical location and half horizontal and half vertical locations respectively, according to $d=(32c_1+c_2+1024)/2048$. Again operator/indicates division with truncation and the finally obtained $1/4$ resolution sub-pixel values d are clipped to lie in the range $[0, 2^n-1]$.
4. Values for the $1/4$ resolution sub-pixels labelled e are calculated as follows. Values for $1/4$ resolution sub-pixels e located at unit horizontal and quarter vertical locations are calculated from the value of the immediately neighbouring pixel at unit horizontal and unit vertical location (pixel A) and the intermediate value c calculated in step (2) for the immediately neighbouring $1/2$ resolution sub-pixel at unit horizontal and half vertical location, according to $e=(32A+c+32)/64$. Values for $1/4$ resolution sub-pixels e located at half horizontal and quarter vertical locations are calculated from the intermediate value b calculated in step (1) for the immediately neighbouring $1/2$ resolution sub-pixel at half horizontal and unit vertical location and the intermediate value c calculated in step (2) for the immediately neighbouring $1/2$ resolution sub-pixel at half horizontal and half vertical location, according to $e=(32b+c+1024)/2048$. Once more, operator/indicates division with truncation and the finally obtained $1/4$ resolution sub-pixel values e are clipped to lie in the range $[0, 2^n-1]$.
5. Values for $1/2$ resolution sub-pixels labelled g are computed using the value of the nearest original pixel A and the intermediate values of the three nearest neighbouring $1/2$ resolution sub-pixels, according to $g=(1024A+32b+32c_1+c_2+2048)/4096$. As before, operator/indicates division with truncation and the finally obtained for $1/4$ resolution sub-pixel values g are clipped to lie in the range $[0, 2^n-1]$.
6. The value for $1/4$ resolution sub-pixel f is interpolated by averaging the values of the 4 closest pixels at unit

13

horizontal and vertical locations, according to $f=(A_1+A_2+A_3+A_4)/4$, where the locations of pixels A_1 , A_2 , A_3 , and A_4 are the four nearest original pixels.

In the decoder, sub-pixel values can be obtained directly by applying 6-tap filters in horizontal and vertical directions. In the case of $1/4$ sub-pixel resolution, referring to FIG. 13a, the filter coefficients applied to pixels and sub-pixels at unit vertical location are $[0, 0, 64, 0, 0, 0]$ for a set of six pixels A , $[1, -5, 52, 20, -5, 1]$ for a set of six sub-pixels d , $[2, -10, 40, 40, -10, 2]$ for a set of six sub-pixels b , and $[1, -5, 20, 52, -5, 1]$ for a set of six sub-pixels c . These filter coefficients are applied to respective sets of pixels or sub-pixels in the same row as the sub-pixel values being interpolated.

After applying the filters in the horizontal and vertical directions, interpolated value c is normalized according to $c=(c+2048)/4096$ and clipped to lie in the range $[0, 2^n-1]$. When a motion vector points to an integer pixel position in either the horizontal or vertical direction, many zero coefficients are used. In a practical implementation of TML6, different branches are used in the software which are optimised for the different sub-pixel cases so that there are no multiplications by zero coefficients.

It should be noted that in TML6, $1/4$ resolution sub-pixel values are obtained directly using the intermediate values referred to above and are not derived from rounded and clipped values for $1/2$ resolution sub-pixels. Therefore, in obtaining the $1/4$ resolution sub-pixel values, it is not necessary to calculate final values for any of the $1/2$ resolution sub-pixels. Specifically, it is not necessary to carry out the truncation and clipping operations associated with the calculation of final values for the $1/2$ resolution sub-pixels. Neither is it necessary to have stored final values for $1/2$ resolution sub-pixels for use in the calculation of the $1/4$ resolution sub-pixel values. Therefore TML6 is computationally less complex than TML5, as fewer truncating and clipping operations are required. However, a disadvantage of TML6 is that high precision arithmetic is required both in the encoder and in the decoder. High precision interpolation requires more silicon area in ASICs and requires more computations in some CPUs. Furthermore, implementation of direct interpolation as specified in TML6 in an on-demand fashion has a high memory requirement. This is an important factor, particularly in embedded devices.

In view of the previously presented discussion, it should be appreciated that due to the different requirements of the video encoder and decoder with regard to sub-pixel interpolation, there exists a significant problem in developing a method of sub-pixel value interpolation capable of providing satisfactory performance in both the encoder and decoder. Furthermore, neither of the current test models (TML5, TML6) described in the foregoing can provide a solution that is optimum for application in both encoder and decoder.

SUMMARY OF THE INVENTION

According to a first aspect of the invention there is provided method of interpolation in video coding in which an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, is interpolated to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $1/2^x$, where x is a positive integer having a maximum value N , the method comprising:

14

- a) when values for sub-pixels at $1/2^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $1/2^{N-1}$ unit vertical locations are required, interpolating such values directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) when values for sub-pixels at $1/2^{N-1}$ unit horizontal and $1/2^{N-1}$ unit vertical locations are required, interpolating such values directly using a choice of a first weighted sum of values for sub-pixels residing at $1/2^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $1/2^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a); and
- c) when a value for a sub-pixel situated at a $1/2^N$ unit horizontal and $1/2^N$ unit vertical location is required, interpolating such a value by taking a weighted average of the value of a first sub-pixel or pixel situated at a $1/2^{N-m}$ unit horizontal and $1/2^{N-n}$ unit vertical location and the value of a second sub-pixel or pixel located at a $1/2^{N-p}$ unit horizontal and $1/2^{N-q}$ unit vertical location, variables m , n , p and q taking integer values in the range 1 to N such that the first and second sub-pixels or pixels are located diagonally with respect to the sub-pixel at $1/2^N$ unit horizontal and $1/2^N$ unit vertical location.

Preferably a first and a second weight are used in the weighted average referred to in (c), the relative magnitudes of the weights being inversely proportional to the (straight-line diagonal) proximity of the first and the second sub-pixel or pixel to the sub-pixel at $1/2^N$ unit horizontal and $1/2^N$ unit vertical location.

In a situation where the first and the second sub-pixel or pixel are symmetrically located with respect to (equidistant from) the sub-pixel at $1/2^N$ unit horizontal and $1/2^N$ unit vertical location, the first and second weights may have equal values.

The first weighted sum of values for sub-pixels residing at $1/2^{N-1}$ unit horizontal and unit vertical locations in step b) may be used when a sub-pixel at $1/2^{N-1}$ unit horizontal and $1/2^{N-1}$ unit vertical location is required.

The second weighted sum of values for sub-pixels residing at unit horizontal and $1/2^{N-1}$ unit vertical locations in step b) may be used when a sub-pixel at $1/2^N$ unit horizontal and $1/2^{N-1}$ unit vertical location is required.

In one embodiment, when values for sub-pixels at $1/2^N$ unit horizontal and unit vertical locations, and $1/2^N$ unit horizontal and $1/2^{N-1}$ unit vertical locations are required, such values are interpolated by taking the average of the values of a first pixel or sub-pixel located at a vertical location corresponding to that of the sub-pixel being calculated and unit horizontal location and a second pixel or sub-pixel located at a vertical location corresponding to that of the sub-pixel being calculated and $1/2^{N-1}$ unit horizontal location.

When values for sub-pixels at unit horizontal and $1/2^N$ unit vertical locations, and $1/2^{N-1}$ unit horizontal and $1/2^N$ unit vertical locations are required, they may be interpolated by taking the average of the values of a first pixel or sub-pixel located at a horizontal location corresponding to that of the sub-pixel being calculated and unit vertical location and a second pixel or sub-pixel located at a horizontal location corresponding to that of the sub-pixel being calculated and $1/2^{N-1}$ unit vertical location.

Values for sub-pixels at $1/2^N$ unit horizontal and $1/2^N$ unit vertical locations may be interpolated by taking the average of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a $1/2^{N-1}$ unit horizontal and $1/2^{N-1}$ unit vertical location.

Values for sub-pixels at $1/2^N$ unit horizontal and $1/2^N$ unit vertical locations may be interpolated by taking the average

15

of values of a sub-pixel located at a $1/2^{N-1}$ unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and $1/2^{N-1}$ unit vertical location.

Values for half of the sub-pixels at $1/2^N$ unit horizontal and $1/2^N$ unit vertical locations may be interpolated by taking the average of a first pair of values of a sub-pixel located at a $1/2^{N-1}$ unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and $1/2^{N-1}$ unit vertical location and values for the other half of the sub-pixels at $1/2^N$ unit horizontal and $1/2^N$ unit vertical locations are interpolated by taking the average of a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a $1/2^{N-1}$ unit horizontal and $1/2^{N-1}$ unit vertical location.

Values for sub-pixels at $1/2^N$ unit horizontal and $1/2^N$ unit vertical locations are alternately interpolated for one such sub-pixel by taking the average of a first pair of values of a sub-pixel located at a $1/2^{N-1}$ unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and $1/2^{N-1}$ unit vertical location and values and for a neighbouring such sub-pixel by taking the average of a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a $1/2^{N-1}$ unit horizontal and $1/2^{N-1}$ unit vertical location.

The sub-pixels at $1/2^N$ unit horizontal and $1/2^N$ unit vertical locations may be alternately interpolated in a horizontal direction.

The sub-pixels at $1/2^N$ unit horizontal and $1/2^N$ unit vertical locations may be alternately interpolated in a horizontal direction.

When values for some sub-pixels at $1/2^N$ unit horizontal and $1/2^N$ unit vertical locations are required, such values may be interpolated by taking the average of a plurality of nearest neighbouring pixels.

At least one of step a) and step b) interpolating sub-pixel values directly using weighted sums may involve the calculation of an intermediate value for the sub-pixel values having a dynamic range greater than the specified dynamic range.

The intermediate value for a sub-pixel having $1/2^{N-1}$ sub-pixel resolution may be used the calculation of a sub-pixel value having $1/2^N$ sub-pixel resolution.

According to a second aspect of the invention, there is provided a method of interpolation in video coding in which an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, is interpolated to generate values for sub-pixels at fractional horizontal and vertical locations, the method comprising:

- a) when values for sub-pixels at half unit horizontal and unit vertical locations, and unit horizontal and half unit vertical locations are required, interpolating such values directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) when values for sub-pixels at half unit horizontal and half unit vertical locations are required, interpolating such values directly using a weighted sum of values for sub-pixels residing at half unit horizontal and unit vertical locations calculated according to step (a); and
- c) when values for sub-pixels at quarter unit horizontal and quarter unit vertical locations are required, interpolating such values by taking the average of at least one pair of a first pair of values of a sub-pixel located at a half unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and half unit vertical location

16

and a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a half unit horizontal and half unit vertical location.

According to a third aspect of the invention, there is provided a method of interpolation in video coding in which an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, is interpolated to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $1/2^x$ where x is a positive integer having a maximum value N, the method comprising:

- a) when values for sub-pixels at $1/2^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $1/2^{N-1}$ unit vertical locations are required, interpolating such values directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) when a value for a sub-pixel at a sub-pixel horizontal and sub-pixel vertical location is required, interpolating such a value directly using a choice of a first weighted sum of values for sub-pixels located at a vertical location corresponding to that of the sub-pixel being calculated and a second weighted sum of values for sub-pixels located at a horizontal location corresponding to that of the sub-pixel being calculated.

The sub-pixels used in the first weighted sum may be sub-pixels residing at $1/2^{N-1}$ unit horizontal and unit vertical locations and the first weighted sum may be used to interpolate a value for a sub-pixel at $1/2^{N-1}$ unit horizontal and $1/2^N$ unit vertical location.

The sub-pixels used in the second weighted sum may be sub-pixels residing at unit horizontal and $1/2^{N-1}$ unit vertical locations and the second weighted sum may be used to interpolate a value for a sub-pixel at $1/2^N$ unit horizontal and $1/2^{N-1}$ unit vertical location.

When values for sub-pixels at $1/2^N$ unit horizontal and $1/2^N$ unit vertical locations are required, they may be interpolated by taking the average of at least one pair of a first pair of values of a sub-pixel located at a $1/2^{N-1}$ unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and $1/2^{N-1}$ unit vertical location and a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a $1/2^{N-1}$ unit horizontal and $1/2^{N-1}$ unit vertical location.

In the foregoing, N may be equal an integer selected from a list consisting of the values 2, 3, and 4.

Sub-pixels at quarter unit horizontal location are to be interpreted as being sub-pixels having as their left-hand nearest neighbour a pixel at unit horizontal location and as their right-hand nearest neighbour a sub-pixel at half unit horizontal location as well as sub-pixels having as their left-hand nearest neighbour a sub-pixel at half unit horizontal location and as their right-hand nearest neighbour a pixel at unit horizontal location. Correspondingly, sub-pixels at quarter unit vertical location are to be interpreted as being sub-pixels having as their upper nearest neighbour a pixel at unit vertical location and as their lower nearest neighbour a sub-pixel at half unit vertical location as well as sub-pixels having as their upper nearest neighbour a sub-pixel at half unit vertical location and as their lower nearest neighbour a pixel at unit vertical location.

The term dynamic range, refers to the range of values which the sub-pixel values and the weighted sums can take.

Preferably changing the dynamic range, whether by extending it or reducing it, means changing the number of bits which are used to represent the dynamic range.

In an embodiment of the invention, the method is applied to an image that is sub-divided into a number of image blocks. Preferably each image block comprises four corners, each corner being defined by a pixel located at a unit horizontal and unit vertical location. Preferably the method is applied to each image block as the block becomes available for sub-pixel value interpolation. Alternatively, sub-pixel value interpolation according to the method of the invention is performed once all image blocks of an image have become available for sub-pixel value interpolation.

Preferably the method is used in video encoding. Preferably the method is used in video decoding.

In one embodiment of the invention, when used in encoding, the method is carried out as before-hand interpolation, in which values for all sub-pixels at half unit locations and values for all sub-pixels at quarter unit locations are calculated and stored before being subsequently used in the determination of a prediction frame during motion predictive coding. In alternative embodiments, the method is carried out as a combination of before-hand and on-demand interpolation. In this case, a certain proportion or category of sub-pixel values is calculated and stored before being used in the determination of a prediction frame and certain other sub-pixel values are calculated only when required during motion predictive coding.

Preferably, when the method is used in decoding, sub-pixels are only interpolated when their need is indicated by a motion vector.

According to a fourth aspect of the invention, there is provided a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the video coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $1/2^x$, where x is a positive integer having a maximum value N , the interpolator being adapted to:

- a) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $1/2^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and $1/2^{N-1}$ unit vertical locations directly using a choice of a first weighted sum of values for sub-pixels residing at $1/2^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $1/2^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a); and
- c) interpolate a value for a sub-pixel situated at a $1/2^N$ unit horizontal and $1/2^N$ unit vertical location by taking a weighted average of the value of a first sub-pixel or pixel situated at a $1/2^{N-m}$ unit horizontal and $1/2^{N-n}$ unit vertical location and the value of a second sub-pixel or pixel located at a $1/2^{N-p}$ unit horizontal and $1/2^{N-q}$ unit vertical location, variables m , n , p and q taking integer values in the range 1 to N such that the first and second sub-pixels or pixels are located diagonally with respect to the sub-pixel at $1/2^N$ unit horizontal and $1/2^N$ vertical location.

The video coder may comprise a video encoder. It may comprise a video decoder. There may be a codec comprising both the video encoder and the video decoder.

According to a fifth aspect of the invention, there is provided a communications terminal comprising a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the video coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $1/2^x$, where x is a positive integer having a maximum value N , the interpolator being adapted to:

- a) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $1/2^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and $1/2^{N-1}$ unit vertical locations directly using a choice of a first weighted sum of values for sub-pixels residing at $1/2^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $1/2^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a); and
- c) interpolate a value for a sub-pixel situated at a $1/2^N$ unit horizontal and $1/2^N$ unit vertical location by taking a weighted average of the value of a first sub-pixel or pixel situated at a $1/2^{N-m}$ unit horizontal and $1/2^{N-n}$ unit vertical location and the value of a second sub-pixel or pixel located at a $1/2^{N-p}$ unit horizontal and $1/2^{N-q}$ unit vertical location, variables m , n , p and q taking integer values in the range 1 to N such that the first and second sub-pixels or pixels are located diagonally with respect to the sub-pixel at $1/2^N$ unit horizontal and $1/2^N$ vertical location.

The communications terminal may comprise a video encoder. It may comprise a video decoder. Preferably, it comprises a video codec comprising a video encoder and a video decoder.

Preferably the communications terminal comprising a user interface, a processor and at least one of a transmitting block and a receiving block, and a video coder according to at least one of the third and fourth aspects of the invention. Preferably the processor controls the operation of the transmitting block and/or the receiving block and the video coder.

According to a sixth aspect of the invention, there is provided a telecommunications system comprising a communications terminal and a network, the telecommunications network and the communications terminal being connected by a communications link over which coded video can be transmitted, the communications terminal comprising a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the video coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $1/2^x$, where x is a positive integer having a maximum value N , the interpolator being adapted to:

- a) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $1/2^{N-1}$

unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;

- b) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and $1/2^{N-1}$ unit vertical locations directly using a choice of a first weighted sum of values for sub-pixels residing at $1/2^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $1/2^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a); and
- c) interpolate a value for a sub-pixel situated at a $1/2^N$ unit horizontal and $1/2^N$ unit vertical location by taking a weighted average of the value of a first sub-pixel or pixel situated at a $1/2^{N-m}$ unit horizontal and $1/2^{N-n}$ unit vertical location and the value of a second sub-pixel or pixel located at a $1/2^{N-p}$ unit horizontal and $1/2^{N-q}$ unit vertical location, variables m, n, p and q taking integer values in the range 1 to N such that the first and second sub-pixels or pixels are located diagonally with respect to the sub-pixel at $1/2^N$ unit horizontal and $1/2^N$ unit vertical location.

Preferably the telecommunications system is a mobile telecommunications system comprising a mobile communications terminal and a wireless network, the connection between the mobile communications terminal and the wireless network being formed by a radio link. Preferably the network enables the communications terminal to communicate with other communications terminals connected to the network over communications links between the other communications terminals and the network.

According to a seventh aspect of the invention, there is provided a telecommunications system comprising a communications terminal and a network, the telecommunications network and the communications terminal being connected by a communications link over which coded video can be transmitted, the network comprising a video coder for coding for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the video coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $1/2^x$, where x is a positive integer having a maximum value N, the interpolator being adapted to:

- a) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $1/2^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and $1/2^{N-1}$ unit vertical locations directly using a choice of a first weighted sum of values for sub-pixels residing at $1/2^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $1/2^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a); and
- c) interpolate a value for a sub-pixel situated at a $1/2^N$ unit horizontal and $1/2^N$ unit vertical location by taking a weighted average of the value of a first sub-pixel or pixel situated at a $1/2^{N-m}$ unit horizontal and $1/2^{N-n}$ unit vertical location and the value of a second sub-pixel or pixel located at a $1/2^{N-p}$ unit horizontal and $1/2^{N-q}$ unit vertical location, variables m, n, p and q taking integer values in the range 1 to N such that the first and second sub-pixels

or pixels are located diagonally with respect to the sub-pixel at $1/2^N$ unit horizontal and $1/2^N$ unit vertical location.

According to an eighth aspect of the invention there is provided a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the resolution of the sub-pixels being determined by a positive integer N, the interpolator being adapted to:

- a) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $1/2^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate a value for a sub-pixel at a sub-pixel horizontal and sub-pixel vertical location is required directly using a choice of a first weighted sum of values for sub-pixels located at a vertical location corresponding to that of the sub-pixel being calculated and a second weighted sum of values for sub-pixels located at a horizontal location corresponding to that of the sub-pixel being calculated.

The interpolator may be further adapted to form the first weighted sum using the values of sub-pixels residing at $1/2^{N-1}$ unit horizontal and unit vertical locations and to use the first weighted sum to interpolate a value for a sub-pixel at $1/2^{N-1}$ unit horizontal and $1/2^N$ unit vertical location.

The interpolator may be further adapted to form the second weighted sum using the values of sub-pixels residing at unit horizontal and $1/2^{N-1}$ unit vertical locations and to use the second weighted sum to interpolate a value for a sub-pixel at $1/2^N$ unit horizontal and $1/2^{N-1}$ unit vertical location.

The interpolator may be further adapted to interpolate values for sub-pixels at $1/2^N$ unit horizontal and $1/2^N$ unit vertical locations by taking the average of at least one pair of a first pair of values of a sub-pixel located at a $1/2^{N-1}$ unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and $1/2^{N-1}$ unit vertical location and a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a $1/2^{N-1}$ unit horizontal and $1/2^{N-1}$ unit vertical location.

According to a ninth aspect of the invention there is provided a communications terminal comprising a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the resolution of the sub-pixels being determined by a positive integer N, the interpolator being adapted to:

- a) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $1/2^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate a value for a sub-pixel at a sub-pixel horizontal and sub-pixel vertical location is required directly using a choice of a first weighted sum of values for sub-pixels located at a vertical location corresponding to that of the sub-pixel being calculated and a second weighted sum of values for sub-pixels located at a horizontal location corresponding to that of the sub-pixel being calculated.

21

According to a tenth aspect of the invention there is provided a telecommunications system comprising a communications terminal and a network, the telecommunications network and the communications terminal being connected by a communications link over which coded video can be transmitted, the communications terminal comprising a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the resolution of the sub-pixels being determined by a positive integer N, the interpolator being adapted to:

- a) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $1/2^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate a value for a sub-pixel at a sub-pixel horizontal and sub-pixel vertical location is required directly using a choice of a first weighted sum of values for sub-pixels located at a vertical location corresponding to that of the sub-pixel being calculated and a second weighted sum of values for sub-pixels located at a horizontal location corresponding to that of the sub-pixel being calculated.

According to an eleventh aspect of the invention there is provided a telecommunications system comprising a communications terminal and a network, the telecommunications network and the communications terminal being connected by a communications link over which coded video can be transmitted, the network comprising a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the resolution of the sub-pixels being determined by a positive integer N, the interpolator being adapted to:

- a) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $1/2^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate a value for a sub-pixel at a sub-pixel horizontal and sub-pixel vertical location is required directly using a choice of a first weighted sum of values for sub-pixels located at a vertical location corresponding to that of the sub-pixel being calculated and a second weighted sum of values for sub-pixels located at a horizontal location corresponding to that of the sub-pixel being calculated.

BRIEF DESCRIPTION OF THE FIGURES

An embodiment of the invention will now be described by way of example only with reference to the accompanying drawings in which:

FIG. 1 shows a video encoder according to the prior art;

FIG. 2 shows a video decoder according to the prior art;

FIG. 3 shows the types of frames used in video encoding;

FIGS. 4a, 4b, and 4c show steps in block-matching;

FIG. 5 illustrates the process of motion estimation to sub-pixel resolution;

22

FIG. 6 shows a terminal device comprising video encoding and decoding equipment in which the method of the invention may be implemented;

FIG. 7 shows a video encoder according an embodiment of the present invention;

FIG. 8 shows a video decoder according to an embodiment of the present invention;

FIG. 11 shows a schematic diagram of a mobile telecommunications network according to an embodiment of the present invention;

FIG. 12a shows a notation for describing pixel and sub-pixel locations specific to TML5;

FIG. 12b shows interpolation of a half resolution sub-pixels;

FIG. 12c shows interpolation of a half resolution sub-pixels;

FIG. 13a shows a notation for describing pixel and sub-pixel locations specific to TML6;

FIG. 13b shows interpolation of a half resolution sub-pixels;

FIG. 13c shows interpolation of a half resolution sub-pixels;

FIG. 14a shows a notation for describing pixel and sub-pixel locations specific to the invention;

FIG. 14b shows interpolation of a half resolution sub-pixels according to the invention;

FIG. 14c shows interpolation of a half resolution sub-pixels according to the invention;

FIG. 15 shows possible choices of diagonal interpolation for sub-pixels;

FIG. 16 shows the half resolution sub-pixel values required to calculate other half resolution sub-pixel values;

FIG. 17a shows the half resolution sub-pixel values that must be calculated in order to interpolate values for quarter resolution sub-pixels in an image block using the interpolation method of TML5;

FIG. 17b shows the half resolution sub-pixel values that must be calculated in order to interpolate values for quarter resolution sub-pixels in an image block using the interpolation method according to the invention;

FIG. 18a shows the numbers of half resolution sub-pixels that must be calculated in order to obtain values for quarter resolution sub-pixels within an image block using the sub-pixel value interpolation method according to TML5;

FIG. 18b shows the numbers of half resolution sub-pixels that must be calculated in order to obtain values for quarter resolution sub-pixels within an image block using the sub-pixel value interpolation method according to the invention;

FIG. 19 shows a numbering scheme for each of the 15 sub-pixel positions;

FIG. 20 shows nomenclature used to describe pixels, half resolution sub-pixels, quarter resolution sub-pixels and eighth resolution sub-pixels

FIG. 21a shows the diagonal direction to be used in the interpolation of each eighth resolution sub-pixel in an embodiment of the invention;

FIG. 21b shows the diagonal direction to be used in the interpolation of each eighth resolution sub-pixel in another embodiment of the invention; and

FIG. 22 shows nomenclature used to describe eighth resolution sub-pixels within an image.

DETAILED DESCRIPTION

FIGS. 1 to 5, 12a, 12b, 12c, 13a, 13b, and 13c have been described in the foregoing.

FIG. 6 presents a terminal device comprising video encoding and decoding equipment which may be adapted to operate in accordance with the present invention. More precisely, the figure illustrates a multimedia terminal **60** implemented according to ITU-T recommendation H.324. The terminal can be regarded as a multimedia transceiver device. It includes elements that capture, encode and multiplex multimedia data streams for transmission via a communications network, as well as elements that receive, de-multiplex, decode and display received multimedia content. ITU-T recommendation H.324 defines the overall operation of the terminal and refers to other recommendations that govern the operation of its various constituent parts. This kind of multimedia terminal can be used in real-time applications such as conversational videotelephony, or non real-time applications such as the retrieval/streaming of video clips, for example from a multimedia content server in the Internet.

In the context of the present invention, it should be appreciated that the H.324 terminal shown in FIG. 6 is only one of a number of alternative multimedia terminal implementations suited to application of the inventive method. It should also be noted that a number of alternatives exist relating to the location and implementation of the terminal equipment. As illustrated in FIG. 6, the multimedia terminal may be located in communications equipment connected to a fixed line telephone network such as an analogue PSTN (Public Switched Telephone Network). In this case the multimedia terminal is equipped with a modem **71**, compliant with ITU-T recommendations V.8, V.34 and optionally V.8bis. Alternatively, the multimedia terminal may be connected to an external modem. The modem enables conversion of the multiplexed digital data and control signals produced by the multimedia terminal into an analogue form suitable for transmission over the PSTN. It further enables the multimedia terminal to receive data and control signals in analogue form from the PSTN and to convert them into a digital data stream that can be demultiplexed and processed in an appropriate manner by the terminal.

An H.324 multimedia terminal may also be implemented in such a way that it can be connected directly to a digital fixed line network, such as an ISDN (Integrated Services Digital Network). In this case the modem **71** is replaced with an ISDN user-network interface. In FIG. 6, this ISDN user-network interface is represented by alternative block **72**.

H.324 multimedia terminals may also be adapted for use in mobile communication applications. If used with a wireless communication link, the modem **71** can be replaced with any appropriate wireless interface, as represented by alternative block **73** in FIG. 6. For example, an H.324/M multimedia terminal can include a radio transceiver enabling connection to the current 2nd generation GSM mobile telephone network, or the proposed 3rd generation UMTS (Universal Mobile Telephone System).

It should be noted that in multimedia terminals designed for two-way communication, that is for transmission and reception of video data, it is advantageous to provide both a video encoder and video decoder implemented according to the present invention. Such an encoder and decoder pair is often implemented as a single combined functional unit, referred to as a 'codec'.

Because a video encoder according to the invention performs motion compensated video encoding to sub-pixel resolution using a specific interpolation scheme and a particular combination of before-hand and on-demand sub-pixel value interpolation, it is generally necessary for a video

decoder of a receiving terminal to be implemented in a manner compatible with the encoder of the transmitting terminal which formed the compressed video data stream. Failure to ensure this compatibility may have an adverse effect on the quality of the motion compensation and the accuracy of reconstructed video frames.

A typical H.324 multimedia terminal will now be described in further detail with reference to FIG. 6.

The multimedia terminal **60** includes a variety of elements referred to as 'terminal equipment'. This includes video, audio and telematic devices, denoted generically by reference numbers **61**, **62** and **63**, respectively. The video equipment **61** may include, for example, a video camera for capturing video images, a monitor for displaying received video content and optional video processing equipment. The audio equipment **62** typically includes a microphone, for example for capturing spoken messages, and a loudspeaker for reproducing received audio content. The audio equipment may also include additional audio processing units. The telematic equipment **63**, may include a data terminal, keyboard, electronic whiteboard or a still image transceiver, such as a fax unit.

The video equipment **61** is coupled to a video codec **65**. The video codec **65** comprises a video encoder and a corresponding video decoder both implemented according to the invention. Such an encoder and a decoder will be described in the following. The video codec **65** is responsible for encoding captured video data in an appropriate form for further transmission over a communications link and decoding compressed video content received from the communications network. In the example illustrated in FIG. 6, the video codec is implemented according to ITU-T recommendation H.263, with appropriate modifications to implement the sub-pixel value interpolation method according to the invention in both the encoder and the decoder of the video codec.

Similarly, the terminal's audio equipment is coupled to an audio codec, denoted in FIG. 6 by reference number **66**. Like the video codec, the audio codec comprises an encoder/decoder pair. It converts audio data captured by the terminal's audio equipment into a form suitable for transmission over the communications link and transforms encoded audio data received from the network back into a form suitable for reproduction, for example on the terminal's loudspeaker. The output of the audio codec is passed to a delay block **67**. This compensates for the delays introduced by the video coding process and thus ensures synchronisation of audio and video content.

The system control block **64** of the multimedia terminal controls end-to-network signalling using an appropriate control protocol (signalling block **68**) to establish a common mode of operation between a transmitting and a receiving terminal. The signalling block **68** exchanges information about the encoding and decoding capabilities of the transmitting and receiving terminals and can be used to enable the various coding modes of the video encoder. The system control block **64** also controls the use of data encryption. Information regarding the type of encryption to be used in data transmission is passed from encryption block **69** to the multiplexer/de-multiplexer (MUX/DMUX unit) **70**.

During data transmission from the multimedia terminal, the MUX/DMUX unit **70** combines encoded and synchronised video and audio streams with data input from the telematic equipment **63** and possible control data, to form a single bit-stream. Information concerning the type of data encryption (if any) to be applied to the bit-stream, provided by encryption block **69**, is used to select an encryption mode.

Correspondingly, when a multiplexed and possibly encrypted multimedia bit-stream is being received, MUX/DMUX unit **70** is responsible for decrypting the bit-stream, dividing it into its constituent multimedia components and passing those components to the appropriate codec(s) and/or terminal equipment for decoding and reproduction.

It should be noted that the functional elements of the multimedia terminal, video encoder, decoder and video codec according to the invention can be implemented as software or dedicated hardware, or a combination of the two. The video encoding and decoding methods according to the invention are particularly suited for implementation in the form of a computer program comprising machine-readable instructions for performing the functional steps of the invention. As such, the encoder and decoder according to the invention may be implemented as software code stored on a storage medium and executed in a computer, such as a personal desktop computer, in order to provide that computer with video encoding and/or decoding functionality.

If the multimedia terminal **60** is a mobile terminal, that is if it is equipped with a radio transceiver **73**, it will be understood by those skilled in the art that it may also comprise additional elements. In one embodiment it comprises a user interface having a display and a keyboard, which enables operation of the multimedia terminal **60** by a user, together with necessary functional blocks including a central processing unit, such as a microprocessor, which controls the blocks responsible for different functions of the multimedia terminal, a random access memory RAM, a read only memory ROM, and a digital camera. The microprocessor's operating instructions, that is program code corresponding to the basic functions of the multimedia terminal **60**, is stored in the read-only memory ROM and can be executed as required by the microprocessor, for example under control of the user. In accordance with the program code, the microprocessor uses the radio transceiver **73** to form a connection with a mobile communication network, enabling the multimedia terminal **60** to transmit information to and receive information from the mobile communication network over a radio path.

The microprocessor monitors the state of the user interface and controls the digital camera. In response to a user command, the microprocessor instructs the camera to record digital images into the RAM. Once an image is captured, or alternatively during the capturing process, the microprocessor segments the image into image segments (for example macroblocks) and uses the encoder to perform motion compensated encoding for the segments in order to generate a compressed image sequence as explained in the foregoing description. A user may command the multimedia terminal **60** to display the captured images on its display or to send the compressed image sequence using the radio transceiver **73** to another multimedia terminal, a video telephone connected to a fixed line network (PSTN) or some other telecommunications device. In a preferred embodiment, transmission of image data is started as soon as the first segment is encoded so that the recipient can start a corresponding decoding process with a minimum delay.

FIG. **11** is a schematic diagram of a mobile telecommunications network according to an embodiment of the invention. Multimedia terminals MS are in communication with base stations BTS by means of a radio link. The base stations BTS are further connected, through a so-called Abis interface, to a base station controller BSC, which controls and manages several base stations.

The entity formed by a number of base stations BTS (typically, by a few tens of base stations) and a single base

station controller BSC, controlling the base stations, is called a base station subsystem BSS. Particularly, the base station controller BSC manages radio communication channels and handovers. The base station controller BSC is also connected, through a so-called A interface, to a mobile services switching centre MSC, which coordinates the formation of connections to and from mobile stations. A further connection is made, through the mobile service switching centre MSC, to outside the mobile communications network. Outside the mobile communications network there may further reside other network(s) connected to the mobile communications network by gateway(s) GTW, for example the Internet or a Public Switched Telephone Network (PSTN). In such an external network, or within the telecommunications network, there may be located video decoding or encoding stations, such as computers PC. In an embodiment of the invention, the mobile telecommunications network comprises a video server VSRVR to provide video data to a MS subscribing to such a service. The video data is compressed using the motion compensated video compression method as described in the foregoing. The video server may function as a gateway to an online video source or it may comprise previously recorded video clips. Typical videotelephony applications may involve, for example, two mobile stations or one mobile station MS and a video telephone connected to the PSTN, a PC connected to the Internet or a H.261 compatible terminal connected either to the Internet or to the PSTN.

FIG. **7** shows a video encoder **700** according to an embodiment the invention. FIG. **8** shows a video decoder **800** according to an embodiment the invention.

The encoder **700** comprises an input **701** for receiving a video signal from a camera or other video source (not shown). It further comprises a DCT transformer **705**, a quantiser **706**, an inverse quantiser **709**, an inverse DCT transformer **710**, combiners **712** and **716**, a before-hand sub-pixel interpolation block **730**, a frame store **740** and an on-demand sub-pixel interpolation block **750**, implemented in combination with motion estimation block **760**. The encoder also comprises a motion field coding block **770** and a motion compensated prediction block **780**. Switches **702** and **714** are operated co-operatively by a control manager **720** to switch the encoder between an INTRA-mode of video encoding and an INTER-mode of video encoding. The encoder **700** also comprises a multiplexer unit (MUX/DMUX) **790** to form a single bit-stream from the various types of information produced by the encoder **700** for further transmission to a remote receiving terminal, or for example for storage on a mass storage medium such as a computer hard drive (not shown).

It should be noted that the presence and implementations of before-hand sub-pixel interpolation block **730** and on-demand sub-pixel value interpolation block **750** in the encoder architecture depend on the way in which the sub-pixel interpolation method according to the invention is applied. In embodiments of the invention in which before-hand sub-pixel value interpolation is not performed, encoder **700** does not comprise before-hand sub-pixel value interpolation block **730**. In other embodiments of the invention, only before-hand sub-pixel interpolation is performed and thus the encoder does not include on-demand sub-pixel value interpolation block **750**. In embodiments in which both before-hand and on-demand sub-pixel value interpolation are performed, both blocks **730** and **750** are present in the encoder **700**.

Operation of the encoder **700** according to the invention will now be described in detail. In the description, it will be

assumed that each frame of uncompressed video, received from the video source at the input **701**, is received and processed on a macroblock-by-macroblock basis, preferably in raster-scan order. It will further be assumed that when the encoding of a new video sequence starts, the first frame of the sequence is encoded in INTRA-mode. Subsequently, the encoder is programmed to code each frame in INTER-format, unless one of the following conditions is met: 1) it is judged that the current frame being coded is so dissimilar from the reference frame used in its prediction that excessive prediction error information is produced; 2) a predefined INTRA frame repetition interval has expired; or 3) feedback is received from a receiving terminal indicating a request for a frame to be coded in INTRA format.

The occurrence of condition 1) is detected by monitoring the output of the combiner **716**. The combiner **716** forms a difference between the current macroblock of the frame being coded and its prediction, produced in the motion compensated prediction block **780**. If a measure of this difference (for example a sum of absolute differences of pixel values) exceeds a predetermined threshold, the combiner **716** informs the control manager **720** via a control line **717** and the control manager **720** operates the switches **702** and **714** so as to switch the encoder **700** into INTRA coding mode. Occurrence of condition 2) is monitored by means of a timer or frame counter implemented in the control manager **720**, in such a way that if the timer expires, or the frame counter reaches a predetermined number of frames, the control manager **720** operates the switches **702** and **714** to switch the encoder into INTRA coding mode. Condition 3) is triggered if the control manager **720** receives a feedback signal from, for example, a receiving terminal, via control line **718** indicating that an INTRA frame refresh is required by the receiving terminal. Such a condition might arise, for example, if a previously transmitted frame were badly corrupted by interference during its transmission, rendering it impossible to decode at the receiver. In this situation, the receiver would issue a request for the next frame to be encoded in INTRA format, thus re-initialising the coding sequence.

It will further be assumed that the encoder and decoder are implemented in such a way as to allow the determination of motion vectors with a spatial resolution of up to quarter-pixel resolution. As will be seen in the following, finer levels of resolution are also possible.

Operation of the encoder **700** in INTRA coding mode will now be described. In INTRA-mode, the control manager **720** operates the switch **702** to accept video input from input line **719**. The video signal input is received macroblock by macroblock from input **701** via the input line **719** and each macroblock of original image pixels is transformed into DCT coefficients by the DCT transformer **705**. The DCT coefficients are then passed to the quantiser **706**, where they are quantised using a quantisation parameter QP. Selection of the quantisation parameter QP is controlled by the control manager **720** via control line **722**. Each DCT transformed and quantised macroblock that makes up the INTRA coded image information **723** of the frame is passed from the quantiser **706** to the MUX/DMUX **790**. The MUX/DMUX **790** combines the INTRA coded image information with possible control information (for example header data, quantisation parameter information, error correction data etc.) to form a single bit-stream of coded image information **725**. Variable length coding (VLC) is used to reduce redundancy of the compressed video bit-stream, as is known to those skilled in the art.

A locally decoded picture is formed in the encoder **700** by passing the data output by the quantiser **706** through inverse quantiser **709** and applying an inverse DCT transform **710** to the inverse-quantised data. The resulting data is then input to the combiner **712**. In INTRA mode, switch **714** is set so that the input to the combiner **712** via the switch **714** is set to zero. In this way, the operation performed by the combiner **712** is equivalent to passing the decoded image data formed by the inverse quantiser **709** and the inverse DCT transform **710** unaltered.

In embodiments of the invention in which before-hand sub-pixel value interpolation is performed, the output from combiner **712** is applied to before-hand sub-pixel interpolation block **730**. The input to the before-hand sub-pixel value interpolation block **730** takes the form of decoded image blocks. In the before-hand sub-pixel value interpolation block **730**, each decoded macroblock is subjected to sub-pixel interpolation in such a way that a predetermined sub-set of sub-pixel resolution sub-pixel values is calculated according to the interpolation method of the invention and is stored together with the decoded pixel values in frame store **740**.

In embodiments in which before-hand sub-pixel interpolation is not performed, before-hand sub-pixel interpolation block is not present in the encoder architecture and the output from combiner **712**, comprising decoded image blocks, is applied directly to frame store **740**.

As subsequent macroblocks of the current frame are received and undergo the previously described coding and decoding steps in blocks **705**, **706**, **709**, **710**, **712**, a decoded version of the INTRA-frame is built up in the frame store **740**. When the last macroblock of the current frame has been INTRA coded and subsequently decoded, the frame store **740** contains a completely decoded frame, available for use as a prediction reference frame in coding a subsequently received video frame in INTER format. In embodiments of the invention in which before-hand sub-pixel value interpolation is performed, the reference frame held in frame store **740** is at least partially interpolated to sub-pixel resolution.

Operation of the encoder **700** in INTER coding mode will now be described. In INTER coding mode, the control manager **720** operates switch **702** to receive its input from line **721**, which comprises the output of the combiner **716**. The combiner **716** forms prediction error information representing the difference between the current macroblock of the frame being coded and its prediction, produced in the motion compensated prediction block **780**. The prediction error information is DCT transformed in block **705** and quantised in block **706** to form a macroblock of DCT transformed and quantised prediction error information. Each macroblock of DCT transformed and quantised prediction error information is passed from the quantiser **706** to the MUX/DMUX unit **790**. The MUX/DMUX unit **790** combines the prediction error information **723** with motion coefficients **724** (described in the following) and control information (for example header data, quantisation parameter information, error correction data etc.) to form a single bit-stream of coded image information, **725**.

Locally decoded prediction error information for the each macroblock of the INTER coded frame is then formed in the encoder **700** by passing the encoded prediction error information **723** output by the quantiser **706** through the inverse quantiser **709** and applying an inverse DCT transform in block **710**. The resulting locally decoded macroblock of prediction error information is then input to combiner **712**. In INTER-mode, switch **714** is set so that the combiner **712** also receives motion predicted macroblocks for the current

INTER frame, produced in the motion compensated prediction block 780. The combiner 712 combines these two pieces of information to produce reconstructed image blocks for the current INTER frame.

As described above when considering INTRA coded frames, in embodiments of the invention in which before-hand sub-pixel value interpolation is performed, the output from combiner 712 is applied to the before-hand sub-pixel interpolation block 730. Thus, the input to the before-hand sub-pixel value interpolation block 730 in INTER coding mode also takes the form of decoded image blocks. In the before-hand sub-pixel value interpolation block 730, each decoded macroblock is subjected to sub-pixel interpolation in such a way that a predetermined sub-set of sub-pixel values is calculated according to the interpolation method of the invention and is stored together with the decoded pixel values in frame store 740. In embodiments in which before-hand sub-pixel interpolation is not performed, before-hand sub-pixel interpolation block is not present in the encoder architecture and the output from combiner 712, comprising decoded image blocks, is applied directly to frame store 740.

As subsequent macroblocks of the video signal are received from the video source and undergo the previously described coding and decoding steps in blocks 705, 706, 709, 710, 712, a decoded version of the INTER frame is built up in the frame store 740. When the last macroblock of the frame has been INTER coded and subsequently decoded, the frame store 740 contains a completely decoded frame, available for use as a prediction reference frame in encoding a subsequently received video frame in INTER format. In embodiments of the invention in which before-hand sub-pixel value interpolation is performed, the reference frame held in frame store 740 is at least partially interpolated to sub-pixel resolution.

Formation of a prediction for a macroblock of the current frame will now be described.

Any frame encoded in INTER format requires a reference frame for motion compensated prediction. This means, inter alia, that when encoding a video sequence, the first frame to be encoded, whether it is the first frame in the sequence, or some other frame, must be encoded in INTRA format. This, in turn, means that when the video encoder 700 is switched into INTER coding mode by control manager 720, a complete reference frame, formed by locally decoding a previously encoded frame, is already available in the frame store 740 of the encoder. In general, the reference frame is formed by locally decoding either an INTRA coded frame or an INTER coded frame.

The first step in forming a prediction for a macroblock of the current frame is performed by motion estimation block 760. The motion estimation block 760 receives the current macroblock of the frame being coded via line 727 and performs a block matching operation in order to identify a region in the reference frame which corresponds substantially with the current macroblock. According to the invention, the block-matching process is performed to sub-pixel resolution in a manner that depends on the implementation of the encoder 700 and the degree of before-hand sub-pixel interpolation performed. However, the basic principle behind the block-matching process is similar in all cases. Specifically, motion estimation block 760 performs block-matching by calculating difference values (e.g. sums of absolute differences) representing the difference in pixel values between the macroblock of the current frame under examination and candidate best-matching regions of pixels/ sub-pixels in the reference frame. A difference value is produced for all possible offsets (e.g. quarter- or one eighth

sub-pixel precision x, y displacements) between the macroblock of the current frame and candidate test region within a predefined search region of the reference frame and motion estimation block 760 determines the smallest calculated difference value. The offset between the macroblock in the current frame and the candidate test region of pixel values/ sub-pixel values in the reference frame that yields the smallest difference value defines the motion vector for the macroblock in question. In certain embodiments of the invention, an initial estimate for the motion vector having unit pixel precision is first determined and then refined to a finer level of sub-pixel precision, as described in the foregoing.

In embodiments of the encoder in which before-hand sub-pixel value interpolation is not performed, all sub-pixel values required in the block matching process are calculated in on-demand sub-pixel value interpolation block 750. Motion estimation block 760 controls on-demand sub-pixel value interpolation block 750 to calculate each sub-pixel value needed in the block-matching process in an on-demand fashion, as and when it is required. In this case, motion estimation block 760 may be implemented so as to perform block-matching as a one-step process, in which case a motion vector with the desired sub-pixel resolution is sought directly, or it may be implemented so as to perform block-matching as a two step process. If the two-step process is adopted, the first step may comprise a search for e.g. a full or half-pixel resolution motion vector and the second step is performed in order to refine the motion vector to the desired sub-pixel resolution. As block matching is an exhaustive process, in which blocks of nxm pixels in the current frame are compared one-by-one with blocks of nxm pixels or sub-pixels in the interpolated reference frame, it should be appreciated that a sub-pixel calculated in an on-demand fashion by the on-demand pixel interpolation block 750 may need to be calculated multiple times as successive difference values are determined. In a video encoder, this approach is not the most efficient possible in terms of computational complexity/burden.

In embodiments of the encoder which use only before-hand sub-pixel value interpolation, block-matching may be performed as a one step process, as all sub-pixel values of the reference frame required to determine a motion vector with the desired sub-pixel resolution are calculated before-hand in block 730 and stored in frame store 740. Thus, they are directly available for use in the block-matching process and can be retrieved as required from frame store 740 by motion estimation block 760. However, even in the case where all sub-pixel values are available from frame store 740, it is still more computationally efficient to perform block-matching as a two-step process, as fewer difference calculations are required. It should be appreciated that while full before-hand sub-pixel value interpolation reduces computational complexity in the encoder, it is not the most efficient approach in terms of memory consumption.

In embodiments of the encoder in which both before-hand and on-demand sub-pixel value interpolation are used, motion estimation block 760 is implemented in such a way that it can retrieve sub-pixel values previously calculated in before-hand sub-pixel value interpolation block 730 and stored in frame store 740 and further control on-demand sub-pixel value interpolation block 750 to calculate any additional sub-pixel values that may be required. The block-matching process may be performed as a one-step or a two-step process. If a two-step implementation is used, before-hand calculated sub-pixel values retrieved from frame store 740 may be used in the first step of the process

and the second step may be implemented so as to use sub-pixel values calculated by on-demand sub-pixel value interpolation block 750. In this case, certain sub-pixel values used in the second step of the block matching process may need to be calculated multiple times as successive comparisons are made, but the number of such duplicate calculations is significantly less than if before-hand sub-pixel value calculation is not used. Furthermore, memory consumption is reduced with respect to embodiments in which only before-hand sub-pixel value interpolation is used.

Once the motion estimation block 760 has produced a motion vector for the macroblock of the current frame under examination, it outputs the motion vector to the motion field coding block 770. Motion field coding block 770 then approximates the motion vector received from motion estimation block 760 using a motion model. The motion model generally comprises a set of basis functions. More specifically, the motion field coding block 770 represents the motion vector as a set of coefficient values (known as motion coefficients) which, when multiplied by the basis functions, form an approximation of the motion vector. The motion coefficients 724 are passed from motion field coding block 770 to motion compensated prediction block 780. Motion compensated prediction block 780 also receives the pixel/sub-pixel values of the best-matching candidate test region of the reference frame identified by motion estimation block 760. In FIG. 7, these values are shown to be passed via line 729 from on-demand sub-pixel interpolation block 750. In alternative embodiments of the invention, the pixel values in question are provided from the motion estimation block 760 itself.

Using the approximate representation of the motion vector generated by motion field coding block 770 and the pixel/sub-pixel values of the best-matching candidate test region, motion compensated prediction block 780 produces a macroblock of predicted pixel values. The macroblock of predicted pixel values represents a prediction for the pixel values of the current macroblock generated from the interpolated reference frame. The macroblock of predicted pixel values is passed to the combiner 716 where it is subtracted from the new current frame in order to produce prediction error information 723 for the macroblock, as described in the foregoing.

The motion coefficients 724 formed by motion field coding block are also passed to the MUX/DMUX unit 790, where they are combined with prediction error information 723 for the macroblock in question and possible control information from control manager 720 to form an encoded video stream 725 for transmission to a receiving terminal.

Operation of a video decoder 800 according to the invention will now be described. Referring to FIG. 8, the decoder 800 comprises a demultiplexing unit (MUX/DMUX) 810, which receives the encoded video stream 725 from the encoder 700 and demultiplexes it, an inverse quantiser 820, an inverse DCT transformer 830, a motion compensated prediction block 840, a frame store 850, a combiner 860, a control manager 870, an output 880, before-hand sub-pixel value interpolation block 845 and on-demand sub-pixel interpolation block 890 associated with the motion compensated prediction block 840. In practice the control manager 870 of the decoder 800 and the control manager 720 of the encoder 700 may be the same processor. This may be the case if the encoder 700 and decoder 800 are part of the same video codec.

FIG. 8 shows an embodiment in which a combination of before-hand and on-demand sub-pixel value interpolation is used in the decoder. In other embodiments, only before-hand

sub-pixel value interpolation is used, in which case decoder 800 does not include on-demand sub-pixel value interpolation block 890. In a preferred embodiment of the invention, no before-hand sub-pixel value interpolation is used in the decoder and therefore before-hand sub-pixel value interpolation block 845 is omitted from the decoder architecture. If both before-hand and on-demand sub-pixel value interpolation are performed, the decoder comprises both blocks 845 and 890.

The control manager 870 controls the operation of the decoder 800 in response to whether an INTRA or an INTER frame is being decoded. An INTRA/INTER trigger control signal, which causes the decoder to switch between decoding modes is derived, for example, from picture type information provided in the header portion of each compressed video frame received from the encoder. The INTRA/INTER trigger control signal is passed to control manager 870 via control line 815, together with other video codec control signals demultiplexed from the encoded video stream 725 by the MUX/DMUX unit 810.

When an INTRA frame is decoded, the encoded video stream 725 is demultiplexed into INTRA coded macroblocks and control information. No motion vectors are included in the encoded video stream 725 for an INTRA coded frame. The decoding process is performed macroblock-by-macroblock. When the encoded information 723 for a macroblock is extracted from video stream 725 by MUX/DMUX unit 810, it is passed to inverse quantiser 820. The control manager controls inverse quantiser 820 to apply a suitable level of inverse quantisation to the macroblock of encoded information, according to control information provided in video stream 725. The inverse quantised macroblock is then inversely transformed in the inverse DCT transformer 830 to form a decoded block of image information. Control manager 870 controls combiner 860 to prevent any reference information being used in the decoding of the INTRA coded macroblock. The decoded block of image information is passed to the video output 880 of the decoder.

In embodiments of the decoder which employ before-hand sub-pixel value interpolation, the decoded block of image information (i.e. pixel values) produced as a result of the inverse quantisation and inverse transform operations performed in blocks 820 and 830 is passed to before-hand sub-pixel value interpolation block 845. Here, sub-pixel value interpolation is performed according to the method of the invention, the degree of before-hand sub-pixel value interpolation applied being determined by the details of the decoder implementation. In embodiments of the invention in which on-demand sub-pixel value interpolation is not performed, before-hand sub-pixel value interpolation block 845 interpolates all sub-pixel values. In embodiments that use a combination of before-hand and on-demand sub-pixel value interpolation, before-hand sub-pixel value interpolation block 845 interpolates a certain sub-set of sub-pixel values. This may comprise, for example, all sub-pixels at half-pixel locations, or a combination of sub-pixels at half-pixel and one quarter-pixel locations. In any case, after before-hand sub-pixel value interpolation, the interpolated sub-pixel values are stored in frame store 850, together with the original decoded pixel values. As subsequent macroblocks are decoded, before-hand interpolated and stored, a decoded frame, at least partially interpolated to sub-pixel resolution is progressively assembled in the frame store 850 and becomes available for use as a reference frame for motion compensated prediction.

In embodiments of the decoder which do not employ before-hand sub-pixel value interpolation, the decoded

block of image information (i.e. pixel values) produced as a result of the inverse quantisation and inverse transform operations performed on the macroblock in blocks **820** and **830** is passed directly to frame store **850**. As subsequent macroblocks are decoded and stored, a decoded frame, having unit pixel resolution is progressively assembled in the frame store **850** and becomes available for use as a reference frame for motion compensated prediction.

When an INTER frame is decoded, the encoded video stream **725** is demultiplexed into encoded prediction error information **723** for each macroblock of the frame, associated motion coefficients **724** and control information. Again, the decoding process is performed macroblock-by-macroblock. When the encoded prediction error information **723** for a macroblock is extracted from the video stream **725** by MUX/DMUX unit **810**, it is passed to inverse quantiser **820**. Control manager **870** controls inverse quantiser **820** to apply a suitable level of inverse quantisation to the macroblock of encoded prediction error information, according to control information received in video stream **725**. The inverse quantised macroblock of prediction error information is then inversely transformed in the inverse DCT transformer **830** to yield decoded prediction error information for the macroblock.

The motion coefficients **724** associated with the macroblock in question are extracted from the video stream **725** by MUX/DMUX unit **810** and passed to motion compensated prediction block **840**, which reconstructs a motion vector for the macroblock using the same motion model as that used to encode the INTER-coded macroblock in encoder **700**. The reconstructed motion vector approximates the motion vector originally determined by motion estimation block **760** of the encoder. The motion compensated prediction block **840** of the decoder uses the reconstructed motion vector to identify the location of a block of pixel/sub-pixel values in a prediction reference frame stored in frame store **850**. The reference frame may be, for example, a previously decoded INTRA frame, or a previously decoded INTER frame. In either case, the block of pixel/sub-pixel values indicated by the reconstructed motion vector, represents the prediction of the macroblock in question.

The reconstructed motion vector may point to any pixel or sub-pixel. If the motion vector indicates that the prediction for the current macroblock is formed from pixel values (i.e. the values of pixels at unit pixel locations), these can simply be retrieved from frame store **850**, as the values in question are obtained directly during the decoding of each frame. If the motion vector indicates that the prediction for the current macroblock is formed from sub-pixel values, these must either be retrieved from frame store **850**, or calculated in on-demand sub-pixel interpolation block **890**. Whether sub-pixel values must be calculated, or can simply be retrieved from the frame store, depends on the degree of before-hand sub-pixel value interpolation used in the decoder.

In embodiments of the decoder that do not employ before-hand sub-pixel value interpolation, the required sub-pixel values are all calculated in on-demand sub-pixel value interpolation block **890**. On the other hand, in embodiments in which all sub-pixel values are interpolated before-hand, motion compensated prediction block **840** can retrieve the required sub-pixel values directly from the frame store **850**. In embodiments that use a combination before-hand and on-demand sub-pixel value interpolation, the action required in order to obtain the required sub-pixel values depends on which sub-pixel values are interpolated before-hand. Taking as an example an embodiment in which all sub-pixel values at half-pixel locations are calculated before-hand, it is evi-

dent that if a reconstructed motion vector for a macroblock points to a pixel at unit location or a sub-pixel at half-pixel location, all the pixel or sub-pixel values required to form the prediction for the macroblock are present in the frame store **850** and can be retrieved from there by motion compensated prediction block **840**. If, however, the motion vector indicates a sub-pixel at a quarter-pixel location, the sub-pixels required to form the prediction for the macroblock are not present in frame store **850** and are therefore calculated in on-demand sub-pixel value interpolation block **890**. In this case, on-demand sub-pixel value interpolation block **890** retrieves any pixel or sub-pixel required to perform the interpolation from frame store **850** and applies the interpolation method described below. Sub-pixel values calculated in on-demand sub-pixel value interpolation block **890** are passed to motion compensated prediction block **840**.

Once a prediction for a macroblock has been obtained, the prediction (that is, a macroblock of predicted pixel values) is passed from motion compensated prediction block **840** to combiner **860** where it combined with the decoded prediction error information for the macroblock to form a reconstructed image block which, in turn, is passed to the video output **880** of the decoder.

It should be appreciated that in practical implementations of encoder **700** and decoder **800**, the extent to which frames are before-hand sub-pixel interpolated, and thus the amount of on-demand sub-pixel value interpolation that is performed, can be chosen according to, or dictated by, the hardware implementation of the video encoder **700**, or the environment in which it is intended to be used. For example, if the memory available to the video encoder is limited, or memory must be reserved for other functions, it is appropriate to limit the amount of before-hand sub-pixel value interpolation that is performed. In other cases, where the microprocessor performing the video encoding operation has limited processing capacity, e.g. the number of operations per second that can be executed is comparatively low, it is more appropriate to restrict the amount of on-demand sub-pixel value interpolation that is performed. In a mobile communications environment, for example, when video encoding and decoding functionality is incorporated in a mobile telephone or similar wireless terminal for communication with a mobile telephone network, both memory and processing power may be limited. In this case a combination of before-hand and on-demand sub-pixel value interpolation may be the best choice to obtain an efficient implementation in the video encoder. In video decoder **800**, use of before-hand sub-pixel value is generally not preferred, as it typically results in the calculation of many sub-pixel values that are not actually used in the decoding process. However, it should be appreciated that although different amounts of before-hand and on-demand interpolation can be used in the encoder and decoder in order to optimise the operation of each, both encoder and decoder can be implemented so as to use the same division between before-hand and on-demand sub-pixel value interpolation.

Although the foregoing description does not describe the construction of bi-directionally predicted frames (B-frames) in the encoder **700** and the decoder **800**, it should be understood that in embodiments of the invention, such a capability may be provided. Provision of such capability is considered within the ability of one skilled in the art.

An encoder **700** or a decoder **800** according to the invention can be realised using hardware or software, or using a suitable combination of both. An encoder or decoder implemented in software may be, for example, a separate program or a software building block that can be used by

35

various programs. In the above description and in the drawings, the functional blocks are represented as separate units, but the functionality of these blocks can be implemented, for example, in one software program unit.

The encoder **700** and decoder **800** can further be combined in order to form a video codec having both encoding and decoding functionality. In addition to being implemented in a multimedia terminal, such a codec may also be implemented in a network. A codec according to the invention may be a computer program or a computer program element, or it may be implemented at least partly using hardware.

The sub-pixel interpolation method used in the encoder **700** and decoder **800** according to the invention now be described in detail. The method will first be introduced at a general conceptual level and then two preferred embodiments will be described. In the first preferred embodiment, sub-pixel value interpolation is performed to $\frac{1}{4}$ pixel resolution and in the second the method is extended to $\frac{1}{8}$ pixel resolution.

It should be noted that interpolation must produce identical values in the encoder and the decoder, but its implementation should be optimized for both entities separately. For example, in an encoder according to the first embodiment of the invention in which sub-pixel value interpolation is performed to $\frac{1}{4}$ pixel resolution, it is most efficient to calculate $\frac{1}{2}$ resolution pixels before-hand and to calculate values for $\frac{1}{4}$ resolution sub-pixels in an on-demand fashion, only when they are needed during motion estimation. This has the effect of limiting memory usage while keeping the computational complexity/burden at an acceptable level. In the decoder, on the other hand, it is advantageous not to pre-calculate any of the sub-pixels. Therefore, it should be appreciated that a preferred embodiment of the decoder does not include before-hand sub-pixel value interpolation block **845** and all sub-pixel value interpolation is performed in on-demand sub-pixel value interpolation block **890**.

In the description of the interpolation method provided below, references are made to the pixel positions depicted in FIG. **14a**. In this figure pixels labelled A represent original pixels (that is, pixels residing at unit horizontal and vertical locations). Pixels labelled with other letters represent sub-pixels that are to be interpolated. The description that follows will adhere to the previously introduced conventions regarding the description of pixel and sub-pixel locations.

Next, the steps required to interpolate all sub-pixel positions are described:

Values for the $\frac{1}{2}$ resolution sub-pixels labelled b are obtained by first calculating an intermediate value b using a Kth order filter, according to:

$$b = \sum_{i=1}^K x_i A_i \quad (9)$$

where x_i is a vector of filter coefficients, A_i is a corresponding vector of original pixel values A situated at unit horizontal and unit vertical locations, and K is an integer which defines the order of the filter. Thus, equation 9 can be re-expressed as:

$$b = x_1 A_1 + x_2 A_2 + x_3 A_3 + \dots + x_{K-1} A_{K-1} + x_K A_K \quad (10)$$

The values of the filter coefficients x_i and the order of the filter K may vary from embodiment to embodiment. Equally, different coefficient values may be used in the calculation of

36

different sub-pixels within an embodiment. In other embodiments, the values of filter coefficients x_i and the order of the filter may depend on which of the $\frac{1}{2}$ resolution b sub-pixels is being interpolated. Pixels A_i are disposed symmetrically with respect to the $\frac{1}{2}$ resolution sub-pixel b being interpolated and are the closest neighbours of that sub-pixel. In the case of the $\frac{1}{2}$ resolution sub-pixel b situated at half horizontal and unit vertical location, pixels A_i are disposed horizontally with respect to b (as shown in FIG. **14b**). If the $\frac{1}{2}$ resolution sub-pixel b situated at unit horizontal and half vertical location is being interpolated, pixels A_i are disposed vertically with respect to b (as shown in FIG. **14c**).

A final value for $\frac{1}{2}$ resolution sub-pixel b is calculated by dividing intermediate value b by a constant scale₁, truncating it to obtain an integer number and clipping the result to lie in the range [0, 2ⁿ-1]. In alternative embodiments of the invention rounding may be performed instead of truncation. Preferably, constant scale₁ is chosen to be equal to the sum of filter coefficients x_i .

A value for the $\frac{1}{2}$ resolution sub-pixel labelled c is also obtained by first calculating an intermediate value c using an Mth order filter, according to:

$$c = \sum_{i=1}^M y_i b_i \quad (11)$$

where y_i is a vector of filter coefficients, b_i is a corresponding vector of intermediate values b_i in the horizontal or vertical direction. i.e.:

$$c = y_1 b_1 + y_2 b_2 + y_3 b_3 + \dots + y_{M-1} b_{M-1} + y_M b_M \quad (12)$$

The values of the filter coefficients y_i and the order of the filter M may vary from embodiment to embodiment. Equally, different coefficient values may be used in the calculation of different sub-pixels within an embodiment. Preferably, the b values are intermediate values for $\frac{1}{2}$ resolution sub-pixels b which are disposed symmetrically with respect to $\frac{1}{2}$ resolution sub-pixel c and are the closest neighbours of sub-pixel c. In an embodiment of the invention, the $\frac{1}{2}$ resolution sub-pixels b are disposed horizontally with respect to sub-pixel c, in an alternative embodiment they are disposed vertically with respect to sub-pixel c.

A final value of $\frac{1}{2}$ resolution sub-pixel c is computed by dividing intermediate value c by a constant scale₂, truncating it to obtain an integer number and clipping the result to lie in the range [0, 2ⁿ-1]. In alternative embodiments of the invention rounding may be performed instead of truncation. Preferably, constant scale₂ is equal to scale₁*scale₁.

It should be noted that the use of intermediate values b in the horizontal direction leads to the same result as using intermediate values b in the vertical direction.

There are two alternatives for interpolating values for the $\frac{1}{4}$ resolution sub-pixels labelled h. Both involve linear interpolation along a diagonal line linking $\frac{1}{2}$ resolution sub-pixels neighbouring the $\frac{1}{4}$ resolution sub-pixel h being interpolated. In a first embodiment, a value for sub-pixel h is calculated by averaging the values of the two $\frac{1}{2}$ resolution sub-pixels b closest to sub-pixel h. In a second embodiment, a value for sub-pixel h is calculated by averaging the values of the closest pixel A and the closest $\frac{1}{2}$ resolution sub-pixel c. It should be appreciated that this provides the possibility of using different combinations of diagonal interpolations to determine the values for sub-pixels h within the confines of different groups of 4 image pixels A. However, it should also

be realised that the same combination should be used in both the encoder and the decoder in order to produce identical interpolation results. FIG. 15 depicts 4 possible choices of diagonal interpolation for sub-pixels h in adjacent groups of 4 pixels within an image. Simulations in the TML environment have verified that both embodiments result in similar compression efficiency. The second embodiment has higher complexity, since calculation of sub-pixel c requires calculation of several intermediate values. Therefore the first embodiment is preferred.

Values for $\frac{1}{4}$ resolution sub-pixels labelled d and g are calculated from the values of their nearest horizontal neighbours using linear interpolation. In other words, a value for $\frac{1}{4}$ resolution sub-pixel d is obtained by averaging values of its nearest horizontal neighbours, original image pixel A and $\frac{1}{2}$ resolution sub-pixel b. Similarly, a value for $\frac{1}{4}$ resolution sub-pixel g is obtained by taking the average of its two nearest horizontal neighbours, $\frac{1}{2}$ resolution sub-pixels b and c.

Values for $\frac{1}{4}$ resolution sub-pixels labelled e, f and i are calculated from the values of their nearest neighbours in the vertical direction using linear interpolation. More specifically, a value for $\frac{1}{4}$ resolution sub-pixel e is obtained by averaging the values of its two nearest vertical neighbours, original image pixel A and $\frac{1}{2}$ resolution sub-pixel b. Similarly, a value for $\frac{1}{4}$ resolution sub-pixel f is obtained by taking the average of its two nearest vertical neighbours, $\frac{1}{2}$ resolution sub-pixels b and c. In an embodiment of the invention, a value for $\frac{1}{4}$ resolution sub-pixel i is obtained in manner identical to that just described in connection with $\frac{1}{4}$ resolution sub-pixel f. However, in an alternative embodiment of the invention, and in common with the H.26 test models TML5 and TML6 previously described, $\frac{1}{4}$ resolution sub-pixel i is determined using the values of the four closest original image pixels, according to $(A_1 + A_2 + A_3 + A_4 + 2)/4$.

It should also be noted that in all cases where an average involving pixel and/or sub-pixel values is determined, the average may be formed in any appropriate manner. For example, the value for $\frac{1}{4}$ resolution sub-pixel d can be defined as $d = (A+b)/2$ or as $d = (A+b+1)/2$. The addition of 1 to the sum of values for pixel A and $\frac{1}{2}$ resolution sub-pixel b has the effect of causing any rounding or truncation operation subsequently applied to round or truncate the value for d to the next highest integer value. This is true for any sum of integer values and may be applied to any of the averaging operations performed according to the method of the invention in order to control rounding or truncation effects.

It should be noted that the sub-pixel value interpolation method according to the invention provides advantages over each of TML5 and TML6.

In contrast to TML5, in which the values of some of the $\frac{1}{4}$ resolution sub-pixels depend on previously interpolated values obtained for other $\frac{1}{4}$ resolution sub-pixels, in the method according to the invention, all $\frac{1}{4}$ resolution sub-pixels are calculated from original image pixels or $\frac{1}{2}$ resolution sub-pixel positions using linear interpolation. Thus, the reduction in precision of those $\frac{1}{4}$ resolution sub-pixel values which occurs in TML5 due to the intermediate truncation and clipping of the other $\frac{1}{4}$ resolution sub-pixels from which they are calculated, does not take place in the method according to the invention. In particular, referring to FIG. 14a, $\frac{1}{4}$ resolution sub-pixels h (and sub-pixel i in one embodiment of the invention) are interpolated diagonally in order to reduce dependency on other $\frac{1}{4}$ -pixels. Furthermore, in the method according to the invention, the

number of calculations (and therefore the number of processor cycles) required to obtain a value for those $\frac{1}{4}$ resolution sub-pixels in the decoder is reduced compared with TML5. Additionally, the calculation of any $\frac{1}{4}$ resolution sub-pixel value requires a number of calculations which is substantially similar to the number of calculations required to determine any other $\frac{1}{4}$ resolution sub-pixel value. More specifically, in a situation where the required $\frac{1}{2}$ resolution sub-pixel values are already available, e.g. they have been calculated before-hand, the number of calculations required to interpolate a $\frac{1}{4}$ resolution sub-pixel value from the pre-calculated $\frac{1}{2}$ resolution sub-pixel values is the same as the number of calculations required to calculate any other $\frac{1}{4}$ resolution sub-pixel value from the available $\frac{1}{2}$ resolution sub-pixel values.

In comparison with TML6, the method according to the invention does not require high precision arithmetic to be used in the calculation of all sub-pixels. Specifically, as all of the $\frac{1}{4}$ resolution sub-pixel values are calculated from original image pixels or $\frac{1}{2}$ resolution sub-pixel values using linear interpolation, lower precision arithmetic can be used in their interpolation. Consequently, in hardware implementations of the inventive method, for example in an ASIC (Application Specific Integrated Circuit), the use of lower precision arithmetic reduces the number of components (e.g. gates) that must be devoted to the calculation of $\frac{1}{4}$ resolution sub-pixel values. This, in turn, reduces the overall area of silicon that must be dedicated to the interpolation function. As the majority of sub-pixels are, in fact, $\frac{1}{4}$ resolution sub-pixels (12 out of the 15 sub-pixels illustrated in FIG. 14a), the advantage provided by the invention in this respect is particularly significant. In software implementations, where sub-pixel interpolation is performed using the standard instruction set of a general purpose CPU (Central Processor Unit) or using a DSP (Digital Signal Processor), a reduction in the precision of the arithmetic required generally leads to an increase in the speed at which the calculations can be performed. This is particularly advantageous in 'low cost' implementations, in which it is desirable to use a general purpose CPU rather than any form of ASIC.

The method according to the invention provides still further advantages compared with TML5. As mentioned previously, in the decoder only 1 out of the 15 sub-pixel positions is required at any given time, namely that which is indicated by received motion vector information. Therefore, it is advantageous if the value of a sub-pixel in any sub-pixel location can be calculated with the minimum number of steps that result in a correctly interpolated value. The method according to the invention provides this capability. As mentioned in the detailed description provided above, $\frac{1}{2}$ resolution sub-pixel c can be interpolated by filtering in either the vertical or the horizontal direction, the same value being obtained for c regardless of whether horizontal or vertical filtering is used. The decoder can therefore take advantage of this property when calculating values for $\frac{1}{4}$ resolution sub-pixels f and g in such a way as to minimise the number of operations required in order to obtain the required values. For example, if the decoder requires a value for $\frac{1}{4}$ resolution sub-pixel f, $\frac{1}{2}$ resolution sub-pixel c should be interpolated in the vertical direction. If a value is required for $\frac{1}{4}$ resolution sub-pixel g, it is advantageous to interpolate a value for c in the horizontal direction. Thus, in general, it can be said that the method according to the invention provides flexibility in the way in which values are derived for certain $\frac{1}{4}$ resolution sub-pixels. No such flexibility is provided in TML5.

Two specific embodiments will now be described in detail. The first represents a preferred embodiment for calculating sub-pixels with up to $\frac{1}{4}$ pixel resolution, while in the second, the method according to the invention is extended to the calculation of values for sub-pixels having up to $\frac{1}{8}$ pixel resolution. For both embodiments a comparison is provided between the computational complexity/ burden resulting from use of the method according to the invention and that which would result from use of the interpolation methods according to TML5 and TML6 in equivalent circumstances.

The preferred embodiment for interpolating sub-pixels at $\frac{1}{4}$ pixel resolution will be described with reference to FIGS. 14a, 14b and 14c. In the following, it will be assumed that all image pixels and final interpolated values for sub-pixels are represented with 8-bits.

Calculation of $\frac{1}{2}$ resolution sub-pixels at i) half unit horizontal and unit vertical locations and ii) unit horizontal and half unit vertical locations.

1. A value for the sub-pixel at half unit horizontal and unit vertical location, that is $\frac{1}{2}$ resolution sub-pixel b in FIG. 14a, is obtained by first calculating intermediate value $b = (A_1 - 5A_2 + 20A_3 + 20A_4 - 5A_5 + A_6)$ using the values of the six pixels (A_1 to A_6) which are situated at unit horizontal and unit vertical locations in either the row or the column of pixels containing b and which are disposed symmetrically about b, as shown in FIGS. 14b and 14c. A final value for $\frac{1}{2}$ resolution sub-pixel b is calculated as $(b+16)/32$ where the operator/denotes division with truncation. The result is clipped to lie in the range $[0, 255]$.

Calculation of $\frac{1}{2}$ resolution sub-pixels at half unit horizontal and half unit vertical locations.

2. A value for the sub-pixel at half unit horizontal and half unit vertical location, that is $\frac{1}{2}$ resolution sub-pixel c in FIG. 14a, is calculated as $c = (b_1 - 5b_2 + 20b_3 + 20b_4 - 5b_5 + b_6 + 512)/1024$ using the intermediate values b for the six closest $\frac{1}{2}$ resolution sub-pixels which are situated in either the row or the column of sub-pixels containing c and which are disposed symmetrically about c. Again, operator/denotes division with truncation and the result is clipped to lie in the range $[0, 255]$. As previously explained, using intermediate values b for $\frac{1}{2}$ resolution sub-pixels b in the horizontal direction leads to the same result as using intermediate values b for $\frac{1}{2}$ resolution sub-pixels b in the vertical direction. Thus, in an encoder according to the invention, the direction for interpolating $\frac{1}{2}$ resolution sub-pixels b can be chosen according to a preferred mode of implementation. In a decoder according to the invention, the direction for interpolating sub-pixels b is chosen according to which, if any, $\frac{1}{4}$ resolution sub-pixels will be interpolated using the result obtained for $\frac{1}{2}$ resolution sub-pixel c.

Calculation of $\frac{1}{4}$ resolution sub-pixels at i) quarter unit horizontal and unit vertical locations; ii) quarter unit horizontal and half unit vertical locations; iii) unit horizontal and quarter unit vertical locations; and iv) half unit horizontal and quarter unit vertical locations.

3. Values for $\frac{1}{4}$ resolution sub-pixels d, situated at quarter unit horizontal and unit vertical locations are calculated according to $d = (A+b)/2$ using the nearest original image pixel A and the closest $\frac{1}{2}$ resolution sub-pixel b in the horizontal direction. Similarly, values for $\frac{1}{4}$ resolution sub-pixels g, situated at quarter unit horizontal and half unit vertical locations are calculated according to $g = (b+c)/2$ using the two nearest $\frac{1}{2}$ reso-

lution sub-pixels in the horizontal direction. In a similar manner, values for $\frac{1}{4}$ resolution sub-pixels e, situated at unit horizontal and quarter unit vertical locations, are calculated according to $e = (A+b)/2$ using the nearest original image pixel A and the closest $\frac{1}{2}$ resolution sub-pixel b in the vertical direction. Values for $\frac{1}{4}$ resolution sub-pixels f, situated at half unit horizontal and quarter unit vertical locations, are determined from $f = (b+c)/2$ using the two nearest $\frac{1}{2}$ resolution sub-pixel in the vertical direction. In all cases, operator/denotes division with truncation.

Calculation of $\frac{1}{4}$ resolution sub-pixels at quarter unit horizontal and quarter unit vertical locations.

4. Values for $\frac{1}{4}$ resolution sub-pixels h, situated at quarter unit horizontal and quarter unit vertical locations are calculated according to $h = (b_1 + b_2)/2$, using the two nearest $\frac{1}{2}$ resolution sub-pixels b in the diagonal direction. Again, operator/denotes division with truncation.
5. A value for the $\frac{1}{4}$ resolution sub-pixel labeled i is computed from $i = (A_1 + A_2 + A_3 + A_4 + 2)/4$ using the four nearest original pixels A. Once more, operator/denotes division with truncation.

An analysis of the computational complexity of the first preferred embodiment of the invention will now be presented.

In the encoder, it is likely that the same sub-pixel values will be calculated multiple times. Therefore, and as previously explained, the complexity of the encoder can be reduced by pre-calculating all sub-pixel values and storing them in memory. However, this solution increases memory usage by a large margin. In the preferred embodiment of the invention, in which motion vector accuracy is $\frac{1}{4}$ pixel resolution in both the horizontal and vertical dimensions, storing pre-calculated sub-pixel values for the whole image requires 16 times the memory required to store the original, non-interpolated image. To reduce memory usage, all $\frac{1}{2}$ resolution sub-pixels can be interpolated before-hand and $\frac{1}{4}$ resolution sub-pixels can be calculated on-demand, that is, only when they are needed. According to the method of the invention, on-demand interpolation of values for $\frac{1}{4}$ resolution sub-pixels only requires linear interpolation from $\frac{1}{2}$ resolution sub-pixels. Four times the original picture memory is required to store the pre-calculated $\frac{1}{2}$ resolution sub-pixels since only 8 bits are necessary to represent them.

However, if the same strategy of pre-calculating all $\frac{1}{2}$ resolution sub-pixels using before-hand interpolation is used in conjunction with the direct interpolation scheme of TML6, the memory requirements increase to 9 times the memory required to store the original non-interpolated image. This results from the fact that a larger number of bits is required to store the high precision intermediate values associated with each $\frac{1}{2}$ resolution sub-pixel in TML6. In addition, the complexity of sub-pixel interpolation during motion estimation is higher in TML6, since scaling and clipping has to be performed for every $\frac{1}{2}$ and $\frac{1}{4}$ sub-pixel location.

In the following, the complexity of the sub-pixel value interpolation method according to the invention, when applied in a video decoder, is compared with that of the interpolation schemes used in TML5 and TML6. Throughout the analysis which follows, it is assumed that in each method the interpolation of any sub-pixel value is performed using only the minimum number of steps required to obtain a correctly interpolated value. It is further assumed that each method is implemented in a block based manner, that is, intermediate values common for all the sub-pixels to be interpolated in a particular $N \times M$ block are calculated only

once. An illustrative example is provided in FIG. 16. Referring to FIG. 16, it can be seen that in order to calculate a 4×4 block of $\frac{1}{2}$ resolution sub-pixels c, a 9×4 block of $\frac{1}{2}$ resolution sub-pixels b is first calculated.

Compared with the sub-pixel value interpolation method used in TML5, the method according to the invention has a lower computational complexity for the following reasons:

1. Unlike the sub-pixel value interpolation scheme used in TML5, according to the method of the invention, a value for $\frac{1}{2}$ resolution sub-pixel c can be obtained by filtering in either the vertical or the horizontal direction. Thus, in order to reduce the number of operations, $\frac{1}{2}$ resolution sub-pixel c can be interpolated in the vertical direction if a value for $\frac{1}{4}$ resolution sub-pixel f is required and in the horizontal direction if a value for $\frac{1}{4}$ resolution sub-pixel g is required. As an example, FIG. 17 shows all the $\frac{1}{2}$ resolution sub-pixel values that must be calculated in order to interpolate values for $\frac{1}{4}$ resolution sub-pixels g in an image block defined by 4×4 original image pixels using the interpolation method of TML5 (FIG. 17a) and using the method according to the invention (FIG. 17b). In this example, the sub-pixel value interpolation method according to TML5 requires a total of $88\frac{1}{2}$ resolution sub-pixels to be interpolated, while the method according to the invention requires the calculation of $72\frac{1}{2}$ resolution sub-pixels. As can be seen from FIG. 17b, according to the invention, $\frac{1}{2}$ resolution sub-pixels c are interpolated in the horizontal direction in order to reduce the number of calculations required.
2. According to the method of the invention, $\frac{1}{4}$ resolution sub-pixel h is calculated by linear interpolation from its two closest neighbouring $\frac{1}{2}$ resolution sub-pixels in the diagonal direction. The respective numbers of $\frac{1}{2}$ resolution sub-pixels that must be calculated in order to obtain values for $\frac{1}{4}$ resolution sub-pixels h within a 4×4 block of original image pixels using the sub-pixel value interpolation method according to TML5 and the method according to the invention are shown in FIGS. 18(a) and 18(b), respectively. Using the method according to TML5 it is necessary to interpolate a total of $56\frac{1}{2}$ resolution sub-pixels, while according to the method of the invention it is necessary to interpolate $40\frac{1}{2}$ resolution sub-pixels.

Table 1 summarizes the decoder complexities of the three sub-pixel value interpolation methods considered here, that according to TML5, the direct interpolation method used in TML6 and the method according to the invention. Complexity is measured in terms of the number of 6-tap filtering and linear interpolation operations performed. It is assumed that Interpolation of $\frac{1}{4}$ resolution sub-pixel i is calculated according to $i = (A_1 + A_2 + A_3 + A_4 + 2)/4$ which is bilinear interpolation and effectively comprises two linear interpolation operations. The operations needed to interpolate sub-pixel values with one 4×4 block of original image pixels are listed for each of the 15 sub-pixel positions which, for convenience of reference, are numbered according to the scheme shown in FIG. 19. Referring to FIG. 19, location 1 is the location of an original image pixel A and locations 2 to 16 are sub-pixel locations. Location 16 is the location of $\frac{1}{4}$ resolution sub-pixel i. In order to compute the average number of operations it has been assumed that the probability of a motion vector pointing to each sub-pixel position is the same. The average complexity is therefore the average of the 15 sums calculated for each sub-pixel location and the single full-pixel location.

TABLE 1

Complexity of $\frac{1}{4}$ resolution sub-pixel interpolation in TML5, TML6 and the method according to the invention.

Location	TML5		TML6		Inventive Method	
	linear.	6-tap	linear.	6-tap	linear.	6-tap
1	0	0	0	0	0	0
3, 9	0	16	0	16	0	16
2, 4, 5, 13	16	16	0	16	16	16
11	0	52	0	52	0	52
7, 15	16	52	0	52	16	52
10, 12	16	68	0	52	16	52
6, 8, 14	48	68	0	52	16	32
16	32	0	32	0	32	0
Average	19	37	2	32	13	28.25

It can be seen from Table 1 that the method according to the invention requires fewer 6-tap filter operations than the sub-pixel value interpolation method according to TML6 and only a few additional linear interpolation operations. Since 6-tap filter operations are much more complex than linear interpolation operations the complexity of the two methods is similar. The sub-pixel value interpolation method according to TML5 has a considerably higher complexity.

The preferred embodiment for interpolating sub-pixels with up to $\frac{1}{8}$ pixel resolution will now be described with reference to FIGS. 20, 21 and 22.

FIG. 20 presents the nomenclature used to describe pixels, $\frac{1}{2}$ resolution sub-pixels, $\frac{1}{4}$ resolution sub-pixels and $\frac{1}{8}$ resolution sub-pixels in this extended application of the method according to the invention.

1. Values for the $\frac{1}{2}$ resolution and $\frac{1}{4}$ resolution sub-pixels labelled b^1 , b^2 and b^3 in FIG. 20 are obtained by first calculating intermediate values $b^1 = (-3A_1 + 12A_2 - 37A_3 + 229A_4 + 71A_5 - 21A_6 + 6A_7 - A_8)$; $b^2 = (-3A_1 + 12A_2 - 39A_3 + 158A_4 + 158A_5 - 39A_6 + 12A_7 - 3A_8)$; and $b^3 = (-A_1 + 6A_2 - 21A_3 + 71A_4 + 229A_5 - 37A_6 + 13A_7 - 3A_8)$ using the values of the eight nearest image pixels (A_1 to A_8) situated at unit horizontal and unit vertical locations in either the row or the column containing b^1 , b^2 and b^3 and disposed symmetrically about $\frac{1}{2}$ resolution sub-pixel b^2 . The asymmetries in the filter coefficients used to obtain intermediate values b^1 and b^3 account for the fact that pixels A_1 to A_8 are not symmetrically located with respect to $\frac{1}{4}$ resolution sub-pixels b^1 and b^3 . Final values for sub-pixels b^i , $i=1, 2, 3$ are calculated according to $b^i = (b^i + 128)/256$ where the operator/ denotes division with truncation. The result is clipped to lie in the range $[0, 255]$.

2. Values for the $\frac{1}{2}$ resolution and $\frac{1}{4}$ resolution sub-pixels labelled c^{1j} , $i, j=1, 2, 3$, are calculated according to $c^{1j} = (-3b_1^j + 12b_2^j - 37b_3^j + 229b_4^j + 71b_5^j - 21b_6^j + 6b_7^j - b_8^j + 32768)/65536$, $c^{2j} = (-3b_1^j + 12b_2^j - 39b_3^j + 158b_4^j + 158b_5^j - 39b_6^j + 12b_7^j - 3b_8^j + 32768)/65536$ and $c^{3j} = (-b_1^j + 6b_2^j - 21b_3^j + 71b_4^j + 229b_5^j - 37b_6^j + 13b_7^j - 3b_8^j + 32768)/65536$ using the intermediate values b^1 , b^2 and b^3 calculated for the eight closest sub-pixels (b_1^j to b_8^j) in the vertical direction, sub-pixels b^j being situated in the column comprising the $\frac{1}{2}$ resolution and $\frac{1}{4}$ resolution sub-pixels c^{1j} being interpolated and disposed symmetrically about the $\frac{1}{2}$ resolution sub-pixel c^{2j} . The asymmetries in the filter coefficients used to obtain values for sub-pixels c^{1j} and c^{3j} account for the fact that sub-pixels b_1^j to b_8^j are not symmetrically located with respect to $\frac{1}{4}$ resolution sub-pixels c^{1j} and c^{3j} . Once more, operator/ denotes division with truncation. Before

the interpolated values for sub-pixels $c^{1/2}$ are stored in the frame memory they are clipped to lie in the range [0, 255]. In an alternative embodiment of the invention, $1/2$ resolution and $1/4$ resolution sub-pixels $c^{1/2}$ are calculated using in an analogous manner using intermediate values b^1 , b^2 and b^3 in the horizontal direction.

3. Values for $1/8$ resolution sub-pixels labelled d are calculated using linear interpolation from the values of their closest neighbouring image pixel, $1/2$ resolution or $1/4$ resolution sub-pixels in the horizontal or vertical direction. For example, upper leftmost $1/8$ resolution sub-pixel d is calculated according to $d=(A+b^1+1)/2$. As before, operator/indicates division with truncation.
4. Values for $1/8$ resolution sub-pixels labelled e and f are calculated using linear interpolation from the values of image pixels, $1/2$ resolution or $1/4$ resolution sub-pixels in the diagonal direction. For example, referring to FIG. 20, upper leftmost pixel $1/8$ resolution sub-pixel e is calculated according to $e=(b^1+b^1+1)/2$. The diagonal direction to be used in the interpolation of each $1/8$ resolution sub-pixel in a first preferred embodiment of the invention, hereinafter referred to as 'preferred method 1', is indicated in FIG. 21(a). Values for $1/8$ resolution sub-pixels labelled g are calculated according to $g=(A+3c^{22}+3)/4$. As always, operator/denotes division with truncation. In an alternative embodiment of the invention, hereinafter referred to as 'preferred method 2', computational complexity is further reduced by interpolating $1/8$ resolution sub-pixels f using linear interpolation from $1/2$ resolution sub-pixels b^2 , that is, according to the relationship $f=(3b^2+b^2+2)/4$. The b^2 sub-pixel which is closer to f is multiplied by 3. The diagonal interpolation scheme used in this alternative embodiment of the invention is depicted in FIG. 21(b). In further alternative embodiments, different diagonal interpolation schemes can be envisaged.

It should be noted that in all cases where an average involving pixel and/or sub-pixel values is used in the determination of $1/8$ resolution sub-pixels, the average may be formed in any appropriate manner. The addition of 1 to the sum of values used in calculating such an average has the effect of causing any rounding or truncation operation subsequently applied to round or truncate the average in question to the next highest integer value. In alternative embodiments of the invention, the addition of 1 is not used.

As in the case of sub-pixel value interpolation to $1/4$ pixel resolution previously described, memory requirements in

the encoder can be reduced by pre-calculating only a part of the sub-pixel values to be interpolated. In the case of sub-pixel value interpolation to $1/8$ pixel resolution, it is advantageous to calculate all $1/2$ resolution and $1/4$ resolution sub-pixels before-hand and to compute values for $1/8$ resolution sub-pixels in an on-demand fashion, only when they are required. When this approach is taken, both the interpolation method according to TML5 and that according to the invention require 16 times the original picture memory to store the $1/2$ resolution and $1/4$ resolution sub-pixel values. However, if the direct interpolation method according to TML6 is used in the same way, intermediate values for the $1/2$ resolution and $1/4$ pixel resolution sub-pixels must be stored. These intermediate values are represented with 32-bit precision and this results in a memory requirement 64 times that for the original, non-interpolated image.

In the following, the complexity of the sub-pixel value interpolation method according to the invention, when applied in a video decoder to calculate values for sub-pixels at up to $1/8$ pixel resolution, is compared with that of the interpolation schemes used in TML5 and TML6. As in the equivalent analysis for $1/4$ pixel resolution sub-pixel value interpolation described above, it is assumed that in each method the interpolation of any sub-pixel value is performed using only the minimum number of steps required to obtain a correctly interpolated value. It is also assumed that each method is implemented in a block based manner, such that intermediate values common for all the sub-pixels to be interpolated in a particular $N \times M$ block are calculated only once.

Table 2 summarizes complexities of the three interpolation methods. Complexity is measured in terms of the number of 8-tap filter and linear interpolation operations performed in each method. The table presents the number of operations required to interpolate each of the $63 \times 1/8$ resolution sub-pixels within one 4×4 block of original image pixels, each sub-pixel location being identified with a corresponding number, as illustrated in FIG. 22. In FIG. 22, location 1 is the location of an original image pixel and locations 2 to 64 are sub-pixel locations. When computing the average number of operations, it has been assumed that the probability of a motion vector pointing to each sub-pixel position is the same. The average complexity is thus the average of the 63 sums calculated for each sub-pixel location and the single full-pixel location.

TABLE 2

Complexity of $1/8$ resolution sub-pixel interpolation in TML5, TML6 and the method according to the invention. (Results shown separately for Preferred Method 1 and Preferred Method 2).

Location	TML5		TML6		Preferred Method 1		Preferred Method 2	
	linear.	8-tap	linear.	8-tap	linear.	8-tap	linear.	8-tap
1	0	0	0	0	0	0	0	0
3, 5, 7, 17, 33, 49	0	16	0	16	0	16	0	16
19, 21, 23, 35, 37, 39, 51, 53, 55	0	60	0	60	0	60	0	60
2, 8, 9, 57	16	16	0	16	16	16	16	16
4, 6, 25, 41	16	32	0	16	16	32	16	32
10, 16, 58, 64	32	76	0	60	16	32	16	32
11, 13, 15, 59, 61, 63	16	60	0	60	16	60	16	60
18, 24, 34, 40, 50, 56	16	76	0	60	16	60	16	60
12, 14, 60, 62	32	120	0	60	16	32	16	32
26, 32, 42, 48	32	108	0	60	16	32	16	32
20, 22, 36, 38, 52, 54	16	120	0	60	16	76	16	76
27, 29, 31, 43, 45, 47	16	76	0	60	16	76	16	76

TABLE 2-continued

Complexity of $\frac{1}{8}$ resolution sub-pixel interpolation in TML5, TML6 and the method according to the invention. (Results shown separately for Preferred Method 1 and Preferred Method 2).								
Location	TML5		TML6		Preferred Method 1		Preferred Method 2	
	linear.	8-tap	linear.	8-tap	linear.	8-tap	linear.	8-tap
28, 30, 44, 46	32	152	0	60	16	60	16	60
Average	64	290.25	0	197.75	48	214.75	48	192.75

As can be seen from Table 2, the number of 8-tap filtering operations performed according to preferred methods 1 and 2 are, respectively, 26% and 34% lower than the number of 8-tap filtering operation performed in the sub-pixel value interpolation method of TML5. The number of linear operations is 25% lower, in both preferred method 1 and preferred method 2, compared with TML5, but this improvement is of lesser importance compared to the reduction in 8-tap filtering operations. It can further be seen that the direct interpolation method used in TML6 has a complexity comparable that of both preferred methods 1 and 2 when used to interpolate values for $\frac{1}{8}$ resolution sub-pixels.

In view of the foregoing description it will be evident to a person skilled in the art that various modifications may be made within the scope of the invention. While a number of preferred embodiments of the invention have been described in detail, it should be apparent that many modifications and variations thereto are possible, all of which fall within the true spirit and scope of the invention.

The invention claimed is:

1. A method for sub-pixel value interpolation to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being arranged in rows and columns, the pixel and sub-pixel locations being representable mathematically within the rectangular bounded region using the co-ordinate notation $K/2^N$, $L/2^N$, K and L being positive integers having respective values between zero and 2^N , N being a positive integer greater than one and representing a particular degree of sub-pixel value interpolation, the method comprising:

interpolating a sub-pixel value for a sub-pixel having co-ordinates with odd values of both K and L, according to a predetermined choice of a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates $1/2$, $1/2$, and a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with even values of both K and L, including zero, situated within a quadrant of the rectangular bounded region defined by corner pixels having co-ordinates $1/2$, $1/2$ and the nearest neighbouring pixel;

interpolating sub-pixel values for sub-pixels having co-ordinates with K equal to an even value and L equal zero and sub-pixels having co-ordinates with K equal to zero and L equal to an even value, used in the interpolation of the sub-pixels having co-ordinates with odd values of both K and L, using weighted sums of the values of pixels located in rows and columns respectively; and

interpolating sub-pixel values for sub-pixels having co-ordinates with even values of both K and L, used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of both K and L,

using a predetermined choice of either a weighted sum of the values of sub-pixels having co-ordinates with K equal to an even value and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, or a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent bounded rectangular regions.

2. A method according to claim 1, comprising using a first and a second weight in the weighted average chosen to interpolate the sub-pixel value for a sub-pixel having co-ordinates with odd values of both K and L, wherein when the chosen weighted average is that involving the value of a nearest neighbouring pixel and the value of the sub-pixel situated at co-ordinates $1/2$, $1/2$, selecting the relative magnitudes of the first and second weights to be inversely proportional to the respective straight-line diagonal distances of the nearest-neighbouring pixel and the sub-pixel situated at co-ordinates $1/2$, $1/2$ to the sub-pixel having co-ordinates with odd values of both K and L whose value is being interpolated.

3. A method according to claim 2, comprising using first and second weights with equal values when the nearest-neighbouring pixel second sub-pixel situated at co-ordinates $1/2$, $1/2$ are equidistant from the sub-pixel having co-ordinates with odd values of both K and L whose value is being interpolated.

4. A method according to claim 1, comprising using a first and a second weight in the weighted average chosen to interpolate the sub-pixel value for a sub-pixel having co-ordinates with odd values of both K and L, wherein when the chosen weighted average is that involving the values of a pair of diagonally-opposed sub-pixels having co-ordinates with even values of both K and L, selecting the relative magnitudes of the first and second weights to be inversely proportional to the respective straight-line diagonal distances of the sub-pixels having co-ordinates with even values of both K and L to the sub-pixel having co-ordinates with odd values of both K and L whose value is being interpolated.

5. A method according to claim 4, comprising using first and second weights with equal values when the sub-pixels having co-ordinates with even values of both K and L are equidistant from the sub-pixel having co-ordinates with odd values of both K and L whose value is being interpolated.

6. A method according to claim 1, comprising interpolating sub-pixel values for sub-pixels having co-ordinates with even values of both K and L, used in interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of both K and L, using a weighted sum of the values of sub-pixels having co-ordinates with K equal to an even value and L equal to zero and the values of sub-pixels

having corresponding co-ordinates in immediately adjacent rectangular bounded regions when the value of a sub-pixel having co-ordinates with K equal to an even value and L equal to an odd value is also required.

7. A method according to claim 1, comprising interpolating sub-pixel values for sub-pixels having co-ordinates with even values of both K and L, used in interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of both K and L, using a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions when the value of a sub-pixel having co-ordinates with K equal to an odd value and L equal to an even value is also required.

8. A method according to claim 1, comprising interpolating sub-pixel values for sub-pixels having co-ordinates with K equal to an odd value O and L equal to an even value E, not including zero or 2^N , by taking an average of the value of a first sub-pixel having co-ordinates with K equal to the even value O-1 and L equal to E and a second sub-pixel having co-ordinates with K equal to the even value O+1 and L equal to E.

9. A method according to claim 1, comprising interpolating a sub-pixel value for a sub-pixel having co-ordinates with K equal to 1 and L equal to zero, by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to zero, and a sub-pixel having co-ordinates with K equal to 2 and L equal to zero.

10. A method according to claim 1, comprising interpolating a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N-1 and L equal to zero, by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to zero, and a sub-pixel having co-ordinates with K equal to 2^N-2 and L equal to zero.

11. A method according to claim 1, comprising interpolating a sub-pixel value for a sub-pixel having co-ordinates with K equal to 1 and L equal to 2^N , by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to 2 and L equal to 2^N .

12. A method according to claim 1, comprising interpolating a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N-1 and L equal to 2^N , by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to 2^N-2 and L equal to 2^N .

13. A method according to claim 1, comprising interpolating sub-pixel values for sub-pixels having co-ordinates with K equal to an even value E, not including zero or 2^N , and L equal to an odd value O, by taking an average of the value of a first sub-pixel having co-ordinates with K equal to E and L equal to the even value O-1 and a second sub-pixel having co-ordinates with K equal to E and L equal to the even value O+1.

14. A method according to claim 1, comprising interpolating a sub-pixel value for a sub-pixel having co-ordinates with K equal to zero and L equal to 1, by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to zero, and a sub-pixel having co-ordinates with K equal to zero and L equal to 2.

15. A method according to claim 1, comprising interpolating a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N and L equal to 1, by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to zero, and a sub-pixel having co-ordinates with K equal to zero and L equal to 2.

16. A method according to claim 1, comprising interpolating a sub-pixel value for a sub-pixel having co-ordinates with K equal to 0 and L equal to 2^N-1 , by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to zero and L equal to 2^N-2 .

17. A method according to claim 1, comprising interpolating a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N and L equal to 2^N-1 , by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to 2^N and L equal to 2^N-2 .

18. A method according to claim 1, comprising interpolating a sub-pixel value for the sub-pixel having co-ordinates with K equal to 2^N-1 and L equal to 2^N-1 , by taking a weighted average of the values of the four corner pixels that define the rectangular bounded region.

19. A method according to claim 1, comprising selecting a value for N from a list consisting of the values 2, 3, and 4.

20. A method for quarter resolution sub-pixel value interpolation to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being arranged in rows and columns, the pixel and sub-pixel locations being representable mathematically within the rectangular bounded region using the co-ordinate notation K/4, L/4, K and L being positive integers having respective values between zero and 4, the method comprising:

interpolating a sub-pixel value for a sub-pixel having co-ordinates with both K and L equal to 1 or 3, according to a predetermined choice of a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates 2/4, 2/4, and a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with both K and L equal to zero, 2 or 4, situated within a quadrant of the rectangular bounded region defined by corner pixels having co-ordinates 2/4, 2/4 and the nearest neighbouring pixel;

interpolating sub-pixel values for sub-pixels having co-ordinates with K equal to 2 and L equal zero and sub-pixels having co-ordinates with K equal to zero and L equal to 2, used in the interpolation of the sub-pixels having co-ordinates with both K and L equal to 1 or 3, using weighted sums of the values of pixels located in rows and columns respectively; and

interpolating a sub-pixel value for the sub-pixel having co-ordinates with both K and L equal to 2, used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with both K and L equal to 1 or 3, using a predetermined choice of either a weighted sum of the values of the sub-pixel having co-ordinates with K equal to 2 and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, or a weighted sum of the value of the sub-pixel having co-ordinates with K equal to zero and L equal to 2 and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions.

21. A method for eighth resolution sub-pixel value interpolation to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being arranged in rows and columns, the

pixel and sub-pixel locations being representable mathematically with the rectangular bounded region using the co-ordinate notation $K/8$, $L/8$, K and L being positive integers having respective values between zero and 8, the method comprising:

interpolating a sub-pixel value for a sub-pixel having co-ordinates with both K and L equal to 1, 3, 5 or 7, according to a predetermined choice of a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates $4/8$, $4/8$, and a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with both K and L equal to zero, 2, 4, 6 or 8, situated within a quadrant of the rectangular bounded region defined by corner pixels having co-ordinates $4/8$, $4/8$ and the nearest neighbouring pixel;

interpolating sub-pixel values for sub-pixels having co-ordinates with K equal to 2, 4 or 6 and L equal zero and sub-pixels having co-ordinates with K equal to zero and L equal to 2, 4 or 6, used in the interpolation of the sub-pixels having co-ordinates with both K and L equal to 1, 3, 5 or 7, using weighted sums of the values of pixels located in rows and columns respectively; and interpolating sub-pixel values for sub-pixels having co-ordinates with both K and L equal to 2, 4 or 6, used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with both K and L equal to 1, 3, 5 or 7, using a predetermined choice of either a weighted sum of the values of sub-pixels having co-ordinates with K equal to 2, 4 or 6 and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions or a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to 2, 4 or 6 and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions.

22. An apparatus for sub-pixel value interpolation, the apparatus being operable to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being arranged in rows and columns, the pixel and sub-pixel locations being representable mathematically within the rectangular bounded region using the co-ordinate notation $K/2^N$, $L/2^N$, K and L being positive integers having respective values between zero and 2^N , N being a positive integer greater than one and representing a particular degree of sub-pixel value interpolation, the apparatus comprising:

circuitry operable to interpolate a sub-pixel value for a sub-pixel having co-ordinates with odd values of both K and L , according to a predetermined choice of a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates $1/2$, $1/2$, and a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with even values of both K and L , including zero, situated within a quadrant of the rectangular bounded region defined by corner pixels having co-ordinates $1/2$, $1/2$ and the nearest neighbouring pixel;

circuitry operable to interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an even value and L equal zero and sub-pixels having co-ordinates with K equal to zero and L equal to an even value, used in the interpolation of the sub-pixels having co-ordinates with odd values of both K and L , using

weighted sums of the values of pixels located in rows and columns respectively; and

circuitry operable to interpolate sub-pixel values for sub-pixels having co-ordinates with even values of both K and L , used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of both K and L , using a predetermined choice of either a weighted sum of the values of sub-pixels having co-ordinates with K equal to an even value and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, or a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent bounded rectangular regions.

23. An apparatus according to claim **22**, wherein the apparatus is operable to use a first and a second weight in the weighted average chosen to interpolate the sub-pixel value for a sub-pixel having co-ordinates with odd values of both K and L , wherein when the chosen weighted average is that involving the value of a nearest neighbouring pixel and the value of the sub-pixel situated at co-ordinates $1/2$, $1/2$, the apparatus is operable to select the relative magnitudes of the first and second weights to be inversely proportional to the respective straight-line diagonal distances of the nearest-neighbouring pixel and the sub-pixel situated at co-ordinates $1/2$, $1/2$ to the sub-pixel having co-ordinates with odd values of both K and L whose value is being interpolated.

24. An apparatus according to claim **23**, wherein the apparatus is operable to use first and second weights with equal values when the nearest-neighbouring pixel and sub-pixel situated at co-ordinates $1/2$, $1/2$ are equidistant from the sub-pixel having co-ordinates with odd values of both K and L whose value is being interpolated.

25. An apparatus according to claim **23**, wherein the apparatus is operable to use first and second weights with equal values when the sub-pixels having co-ordinates with even values of both K and L are equidistant from the sub-pixel having co-ordinates with odd values of both K and L whose value is being interpolated.

26. An apparatus according to claim **22**, wherein the apparatus is operable to use a first and a second weight in the weighted average chosen to interpolate the sub-pixel value for a sub-pixel having co-ordinates with odd values of both K and L , wherein when the chosen weighted average is that involving the values of a pair of diagonally-opposed sub-pixels having co-ordinates with even values of both K and L , the apparatus is operable to select the relative magnitudes of the first and second weights to be inversely proportional to the respective straight-line diagonal distances of the sub-pixels having co-ordinates with even values of both K and L to the sub-pixel having co-ordinates with odd values of both K and L whose value is being interpolated.

27. A communications terminal comprising a video encoder according to claim **23**.

28. A communications terminal comprising a video decoder according to claim **26**.

29. An apparatus according to claim **22**, wherein the apparatus is operable to interpolate sub-pixel values for sub-pixels having co-ordinates with even values of both K and L , used in interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of both K and L , using a weighted sum of the values of sub-pixels having co-ordinates with K equal to an even value and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular

51

bounded regions when the value of a sub-pixel having co-ordinates with K equal to an even value and L equal to an odd value is also required.

30. An apparatus according to claim 22, wherein the apparatus is operable to interpolate sub-pixel values for sub-pixels having co-ordinates with even values of both K and L, used in interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of both K and L, using a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions when the value of a sub-pixel having co-ordinates with K equal to an odd value and L equal to an even value is also required.

31. An apparatus according to claim 22, wherein the apparatus is operable to interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an odd value O and L equal to an even value E, not including zero or 2^N , by taking an average of the value of a first sub-pixel having co-ordinates with K equal to the even value O-1 and L equal to E and a second sub-pixel having co-ordinates with K equal to the even value O+1 and L equal to E.

32. An apparatus according to claim 22, wherein the apparatus is operable to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 1 and L equal to zero, by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to zero, and a sub-pixel having co-ordinates with K equal to 2 and L equal to zero.

33. An apparatus according to claim 22, wherein the apparatus is operable to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N-1 and L equal to zero, by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to zero, and a sub-pixel having co-ordinates with K equal to 2^N-2 and L equal to zero.

34. An apparatus according to claim 22, wherein the apparatus is operable to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 1 and L equal to 2^N , by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to 2 and L equal to zero.

35. An apparatus according to claim 22, wherein the apparatus is operable to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N-1 and L equal to 2^N , by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to 2^N-2 and L equal to 2^N .

36. An apparatus according to claim 22, wherein the apparatus is operable to interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an even value E, not including zero or 2^N , and L equal to an odd value O, by taking an average of the value of a first sub-pixel having co-ordinates with K equal to E and L equal to the even value O-1 and a second sub-pixel having co-ordinates with K equal to E and L equal to the even value O+1.

52

37. An apparatus according to claim 22, wherein the apparatus is operable to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to zero and L equal to 1, by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to zero, and a sub-pixel having co-ordinates with K equal to zero and L equal to 2.

38. An apparatus according to claim 22, wherein the apparatus is operable to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N and L equal to 1, by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to zero, and a sub-pixel having co-ordinates with K equal to zero and L equal to 2.

39. An apparatus according to claim 22, wherein the apparatus is operable to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 0 and L equal to 2^N-1 , by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to zero and L equal to 2^N-2 .

40. An apparatus according to claim 22, wherein the apparatus is operable to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N and L equal to 2^N-1 , by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to 2^N and L equal to 2^N-2 .

41. An apparatus according to claim 22, wherein the apparatus is operable to interpolate a sub-pixel value for the sub-pixel having co-ordinates with K equal to 2^N-1 and L equal to 2^N-1 , by taking a weighted average of the values of the four corner pixels that define the rectangular bounded region.

42. An apparatus according to claim 22, wherein N is set equal to 2.

43. An apparatus according to claim 22, wherein N is set equal to 3.

44. A video encoder comprising an apparatus for sub-pixel value interpolation according to claim 22.

45. A still image encoder comprising an apparatus for sub-pixel value interpolation according to claim 22.

46. A video decoder comprising an apparatus for sub-pixel value interpolation according to claim 22.

47. A communications terminal comprising a still image encoder according to claim 45.

48. A still image decoder comprising an apparatus for sub-pixel value interpolation according to claim 22.

49. A codec comprising a video encoder according to claim 44 and a video decoder according to claim 46.

50. A communications terminal comprising a still image decoder according to claim 48.

51. A codec comprising a still image encoder according to claim 45 and a still image decoder according to claim 48.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 7,280,599 B2
APPLICATION NO. : 11/090717
DATED : October 9, 2007
INVENTOR(S) : Karczewicz et al.

Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In column 46, line 39, in claim 3, delete "second" and insert --and--, therefor.

In column 47, line 47, in claim 12, delete "2_N" and insert --2^N--, therefor.

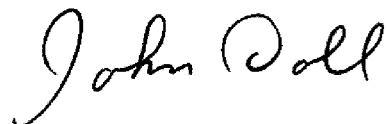
In column 51, line 44, in claim 34, delete "zero" and insert --2^N--, therefor.

In column 50, line 56, in claim 27, delete "23" and insert --44--, therefor.

In column 50, line 58, in claim 28, delete "26" and insert --46--, therefor.

Signed and Sealed this

Fourteenth Day of July, 2009

A handwritten signature in cursive script that reads "John Doll".

JOHN DOLL
Acting Director of the United States Patent and Trademark Office

EXHIBIT 9



US008036273B2

(12) **United States Patent**
Karczewicz et al.

(10) **Patent No.:** **US 8,036,273 B2**
(45) **Date of Patent:** ***Oct. 11, 2011**

(54) **METHOD FOR SUB-PIXEL VALUE INTERPOLATION**

FOREIGN PATENT DOCUMENTS

EP	0576290 A2	12/1993
GB	2205707	12/1988
WO	9904574	1/1999
WO	9956247	11/1999

(75) Inventors: **Marta Karczewicz**, Irving, TX (US);
Antti Hallapuro, Tampere, FL (US)

OTHER PUBLICATIONS

(73) Assignee: **Nokia Corporation**, Espoo (FI)

Two-Dimensional Adaptive Interpolation for Subsampled HDTV Signals; vol. 12, No. 4, Journal of China Institute of Communications, Jul. 1991; pp. 8-12.

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(Continued)

This patent is subject to a terminal disclaimer.

Primary Examiner — Shawn An

(74) Attorney, Agent, or Firm — Perman & Green, LLP

(21) Appl. No.: **11/839,205**

(57) **ABSTRACT**

(22) Filed: **Aug. 15, 2007**

A method of interpolation in video coding in which an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, is interpolated to generate values for sub-pixels at fractional horizontal and vertical locations, the method comprising:

(65) **Prior Publication Data**

US 2008/0069203 A1 Mar. 20, 2008

Related U.S. Application Data

(63) Continuation of application No. 11/090,717, filed on Mar. 25, 2005, now Pat. No. 7,280,599, which is a continuation of application No. 09/954,608, filed on Sep. 17, 2001, now Pat. No. 6,950,469.

a) when values for sub-pixels at half unit horizontal and unit vertical locations, and unit horizontal and half unit vertical locations are required, interpolating such values directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;

(51) **Int. Cl.**
H04B 1/66 (2006.01)

(52) **U.S. Cl.** **375/240.17**; 375/240.16; 375/240.25;
375/240.26; 375/240.12; 382/238; 382/236;
382/233; 382/235

b) when values for sub-pixels at half unit horizontal and half unit vertical locations are required, interpolating such values directly using a weighted sum of values for sub-pixels residing at half unit horizontal and unit vertical locations calculated according to step (a); and

(58) **Field of Classification Search** 375/240.17,
375/240.16, 240.12, 240.25, 240.26; 382/238,
382/236, 233, 235

See application file for complete search history.

c) when values for sub-pixels at quarter unit horizontal and quarter unit vertical locations are required, interpolating such values by taking the average of at least one pair of a first pair of values of a sub-pixel located at a half unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and half unit vertical location and a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a half unit horizontal and half unit vertical location.

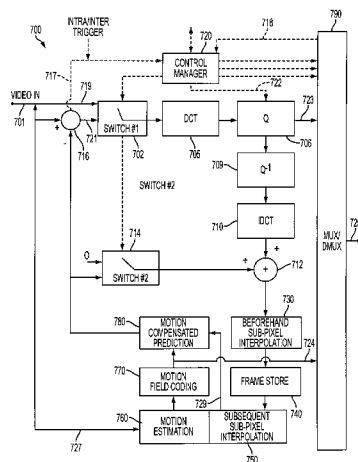
(56) **References Cited**

U.S. PATENT DOCUMENTS

4,937,666 A * 6/1990 Yang 348/413.1

(Continued)

83 Claims, 28 Drawing Sheets



U.S. PATENT DOCUMENTS

5,337,088	A	8/1994	Honjo	
5,452,377	A	9/1995	Igarashi	
5,488,419	A	1/1996	Hui et al.	
5,521,642	A	5/1996	Park	
5,568,597	A	10/1996	Nakayama et al.	
5,570,436	A	10/1996	Fukushima et al.	
5,844,616	A	12/1998	Collet et al.	
5,901,248	A	5/1999	Fandrianto et al.	
6,219,464	B1	4/2001	Greggain et al.	
6,252,576	B1	6/2001	Nottingham	
6,693,960	B2 *	2/2004	Ito et al.	375/240.11
7,280,599	B2 *	10/2007	Karczewicz et al.	375/240.17
2002/0064229	A1	5/2002	Nakaya	

OTHER PUBLICATIONS

Interpolative Prediction Coding Method for HDTV Signals; vol. 32, No. 1, Communication Engineering, Feb. 1992; pp. 81-88.
Wedi, Thomas, "Complexity Reduced Motion Compensated Prediction with 1/8-pel Displacement Vector Resolution," ITU-Telecom-

munications Standardization Sector Study Group 16 Question 6 Video Coding Experts Group (VCEG) Twelfth Meeting: Eibsee, Germany, Jan. 9-12, 2001, [VCEG-L20].

Bjontegaard, Gisle, "H.26L Test Model Long Term Number 6 (TML-6) draft0," ITU-Telecommunications Standardization Sector Study Group 16 Question 6 Video Coding Experts Group (VCEG) 12th meeting: Eibsee, Germany, Jan. 9-12, 2001, [VCEG-L45], p.15.

Wedi, Thomas, "Direct Interpolation Filters in TML-6," ITU-Telecommunications Standardization Sector Study Group 16 Question 6 Video Coding Experts Group (VCEG) Thirteenth Meeting: Austin, Texas, USA, Apr. 2-4, 2001, [VCEG-M44].

"Notice of Reasons for Rejection," JP App. No. 2003-529764, mailed Mar. 3, 2009.

Mehran Mashfeghi, "Directional Interpolation for Magnetic Resonance Angiography Data", IEEE Transactions on Medical Imaging, vol. 12, No. 2, Jun. 1993.

European Office Action dated Jul. 26, 2010.

* cited by examiner

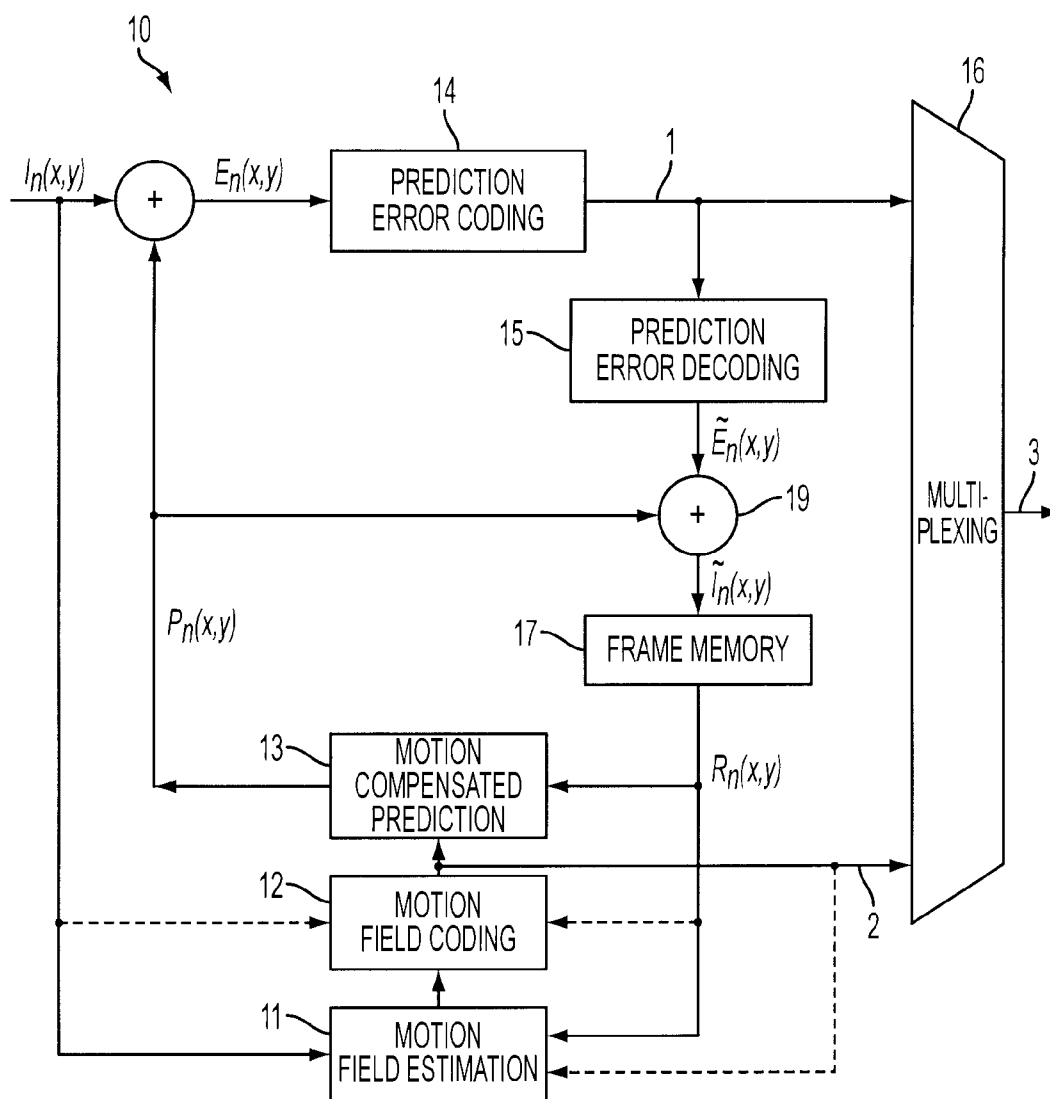


FIG. 1
PRIOR ART

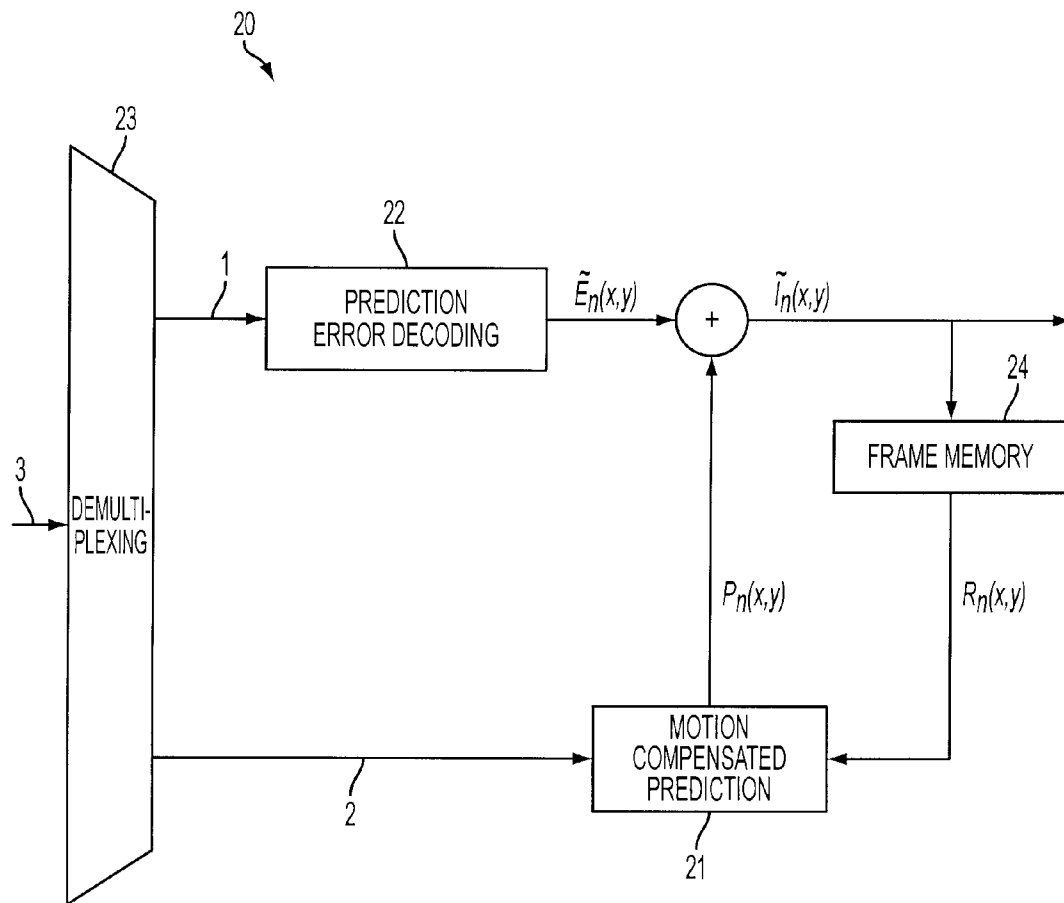


FIG. 2
PRIOR ART

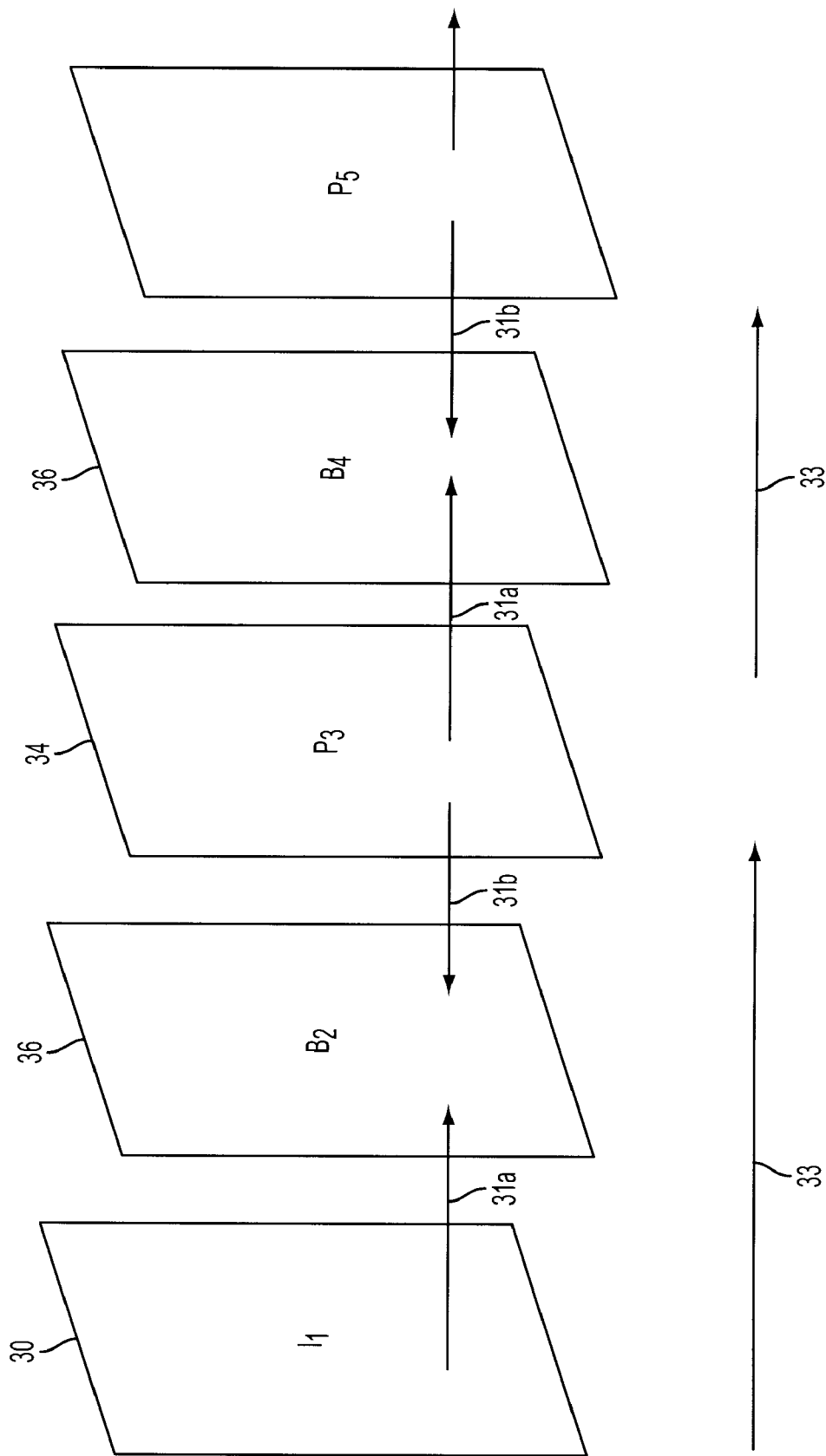


FIG. 3

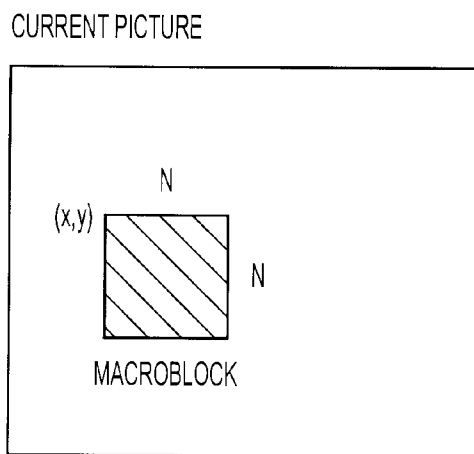


FIG. 4a

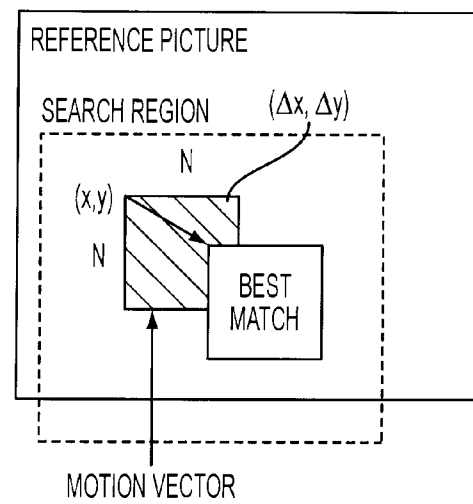
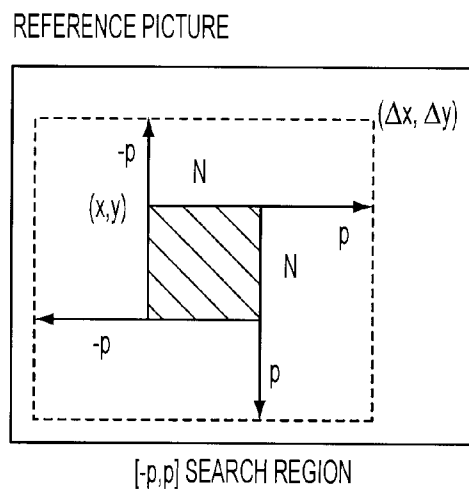


FIG. 4b



[-p, p] SEARCH REGION

FIG. 4c

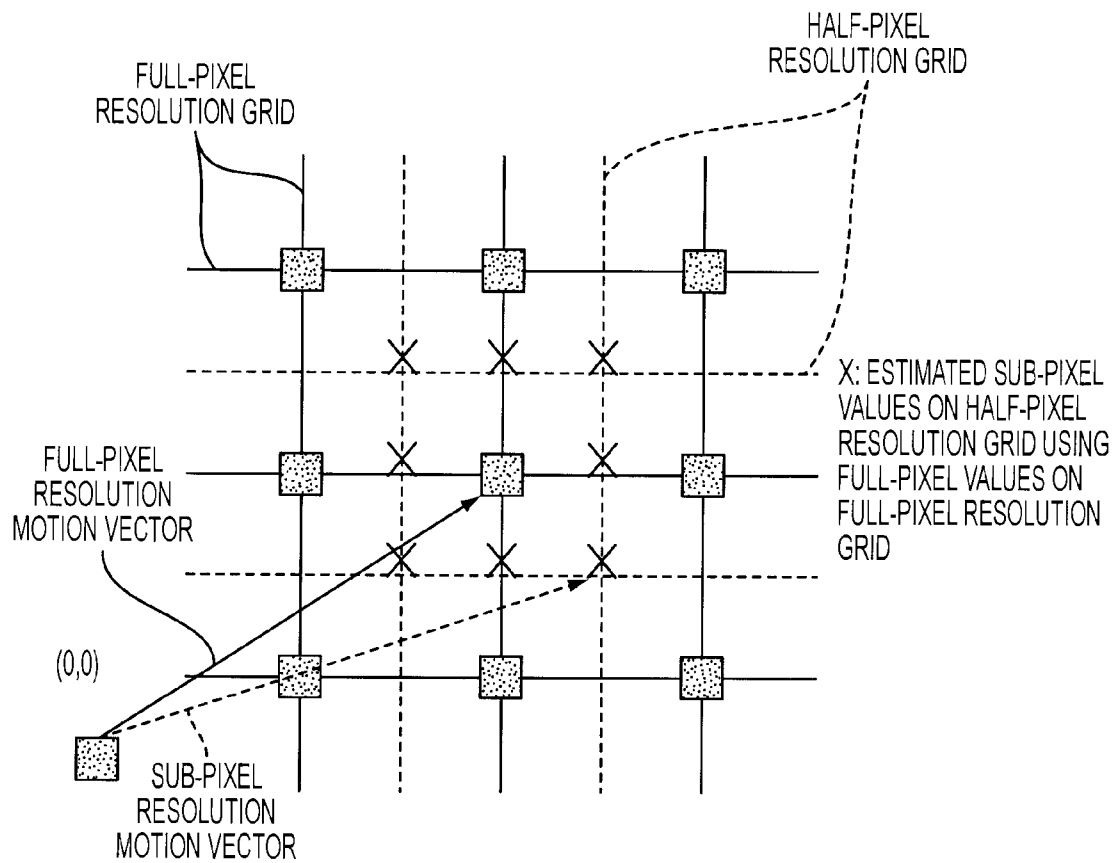


FIG. 5

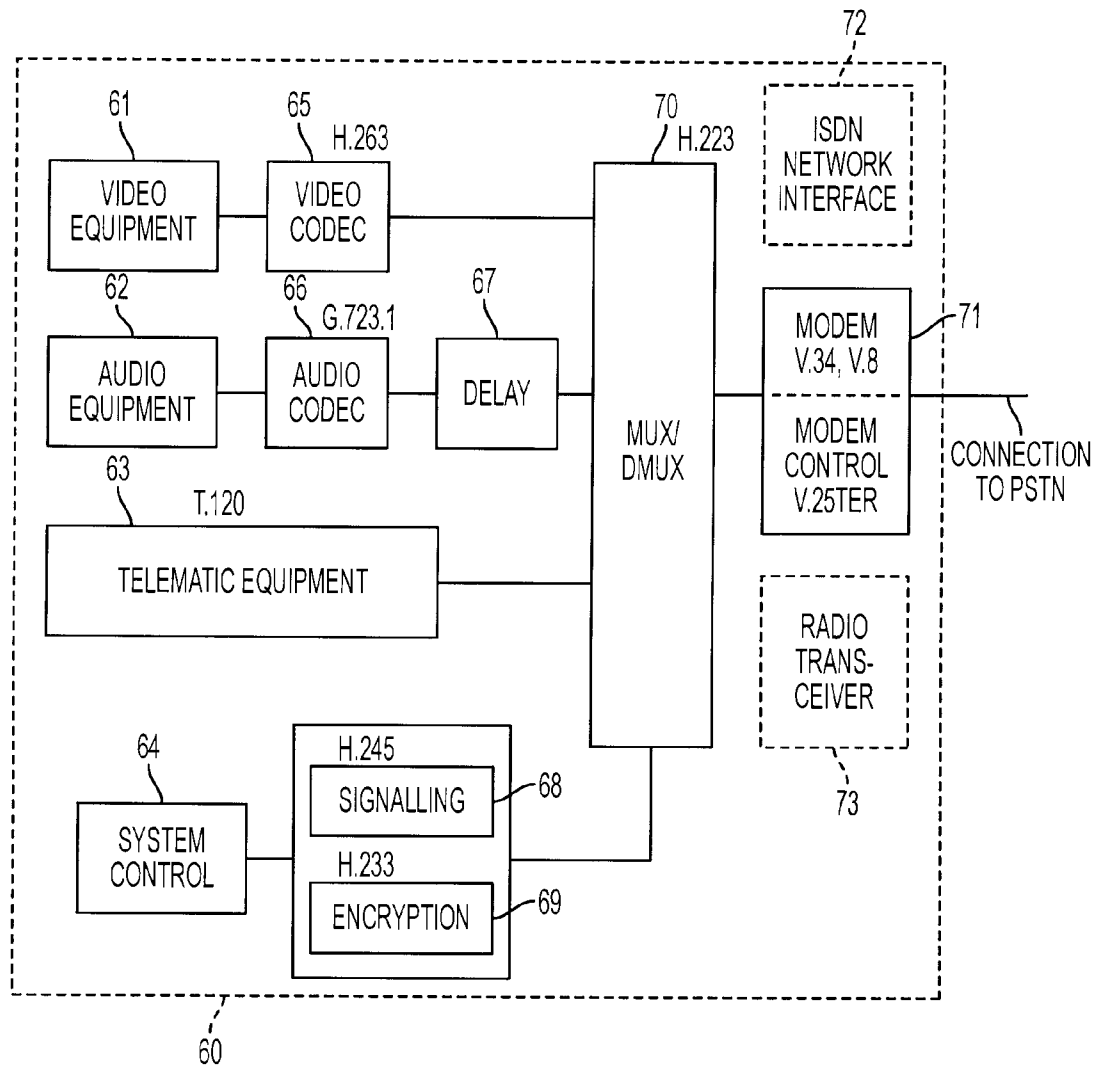


FIG. 6

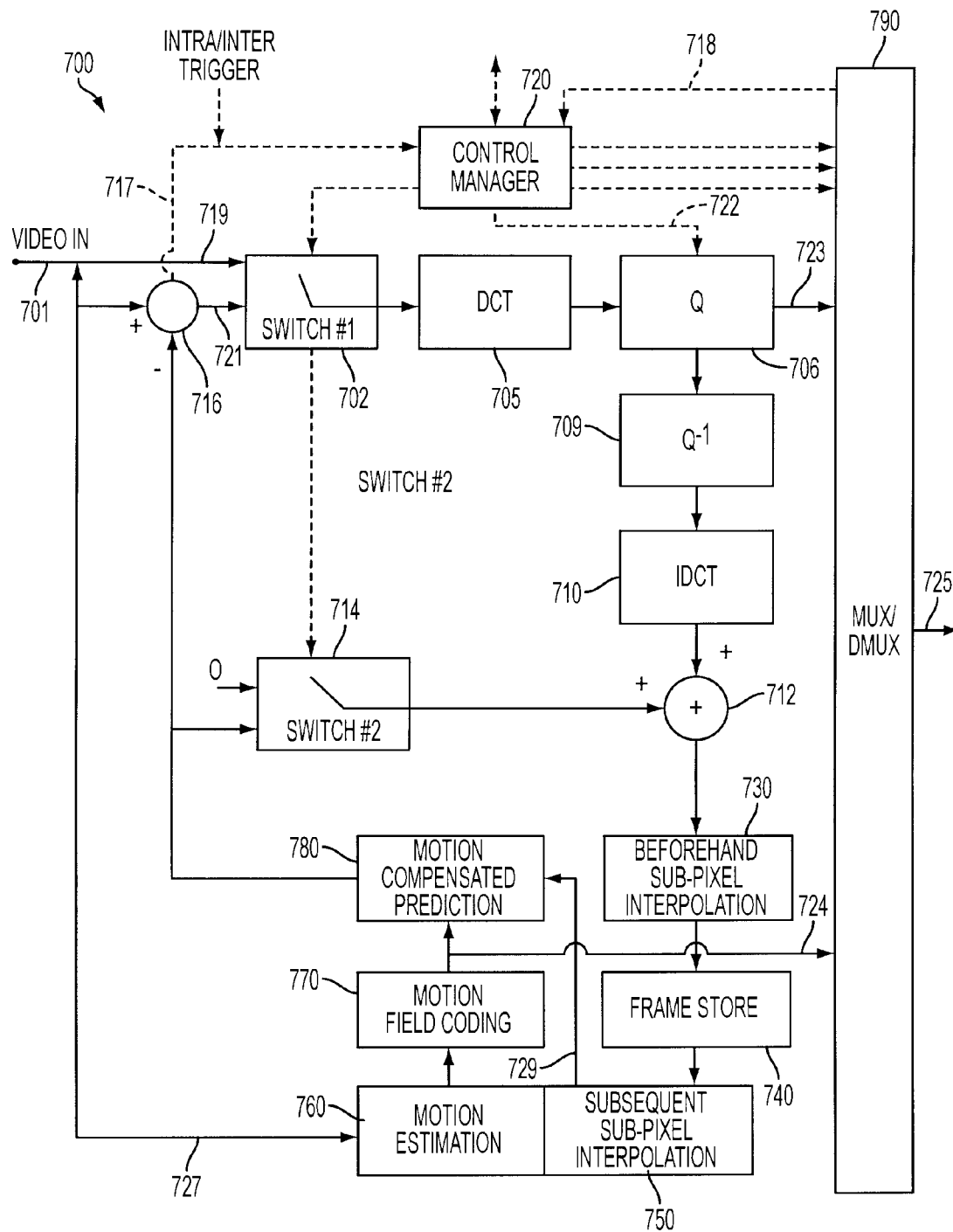


FIG. 7

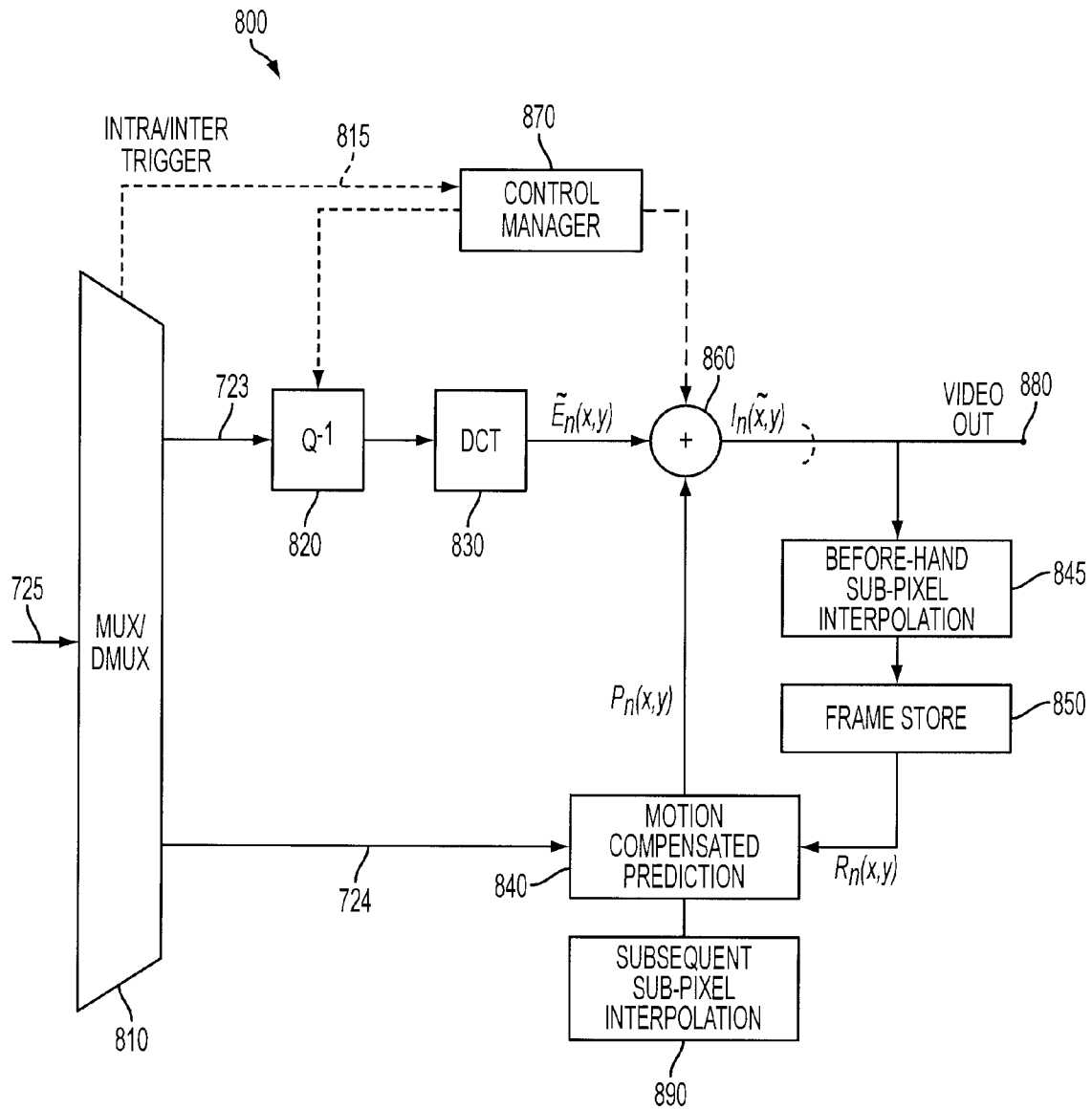


FIG. 8

FIGURE INTENTIONALLY OMITTED

FIG. 9

FIGURE INTENTIONALLY OMITTED

FIG. 10

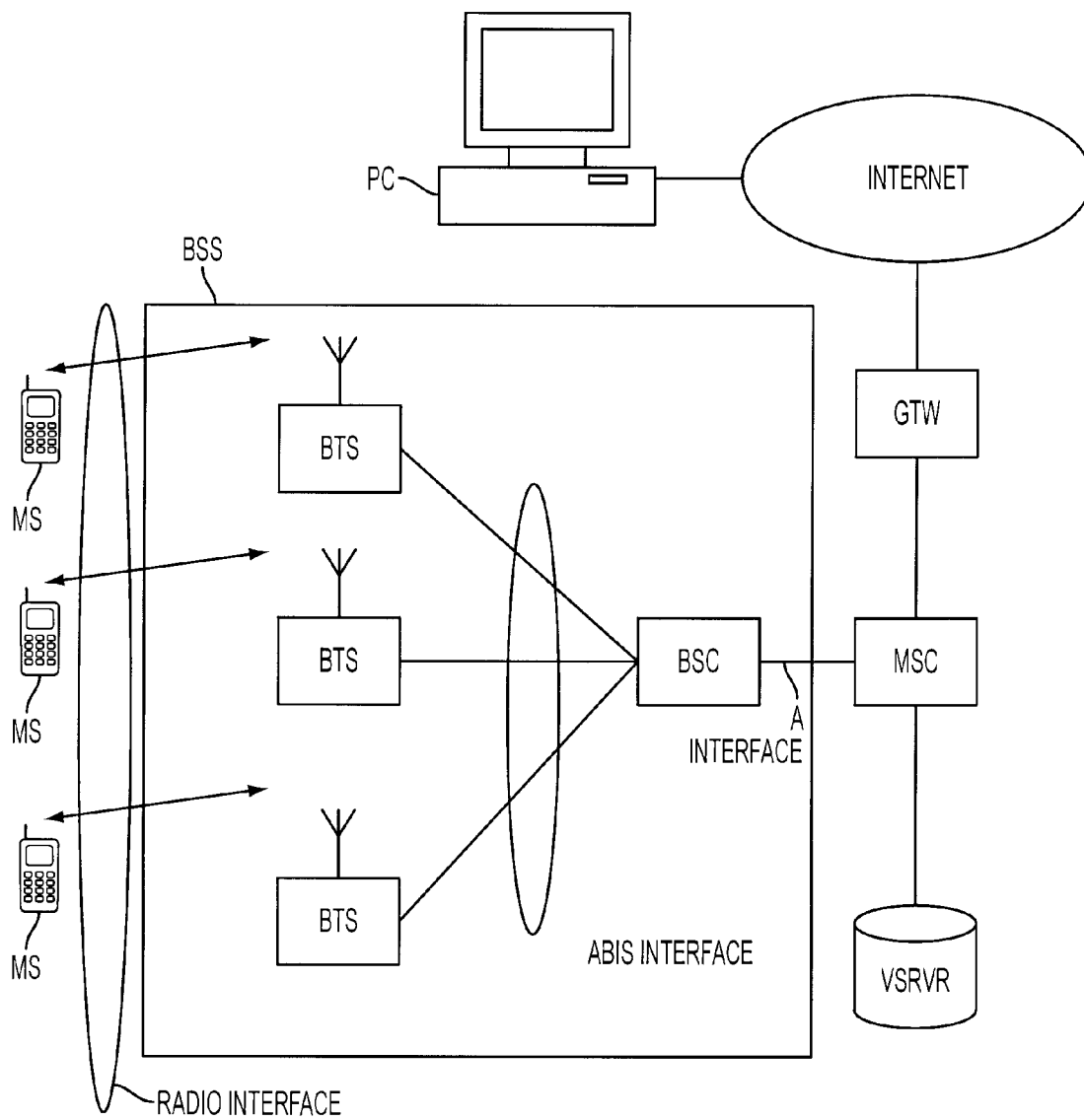


FIG. 11

A	d	b	d	A
e	e	e	e	
c	d	c	d	
e	e	e	f	
A				A

FIG. 12a

A₆

A₅

A₄

d

b

d

A₃

A₂

A₁

e

e

e

e

d

c

d

c

f

e

e

e

A

A

FIG. 12b

A₁

b₁

A₂

b₂

A₃

d

b₃

d

A

e

e

e

e

c

d

c

d

e

e

e

f

A₄

b₄

A

A₅

b₅

A₆

b₆

FIG. 12c

A	d	b	d	A
e	g	e	g	
c	d	c	d	
e	g	e	f	
A				A

FIG. 13a

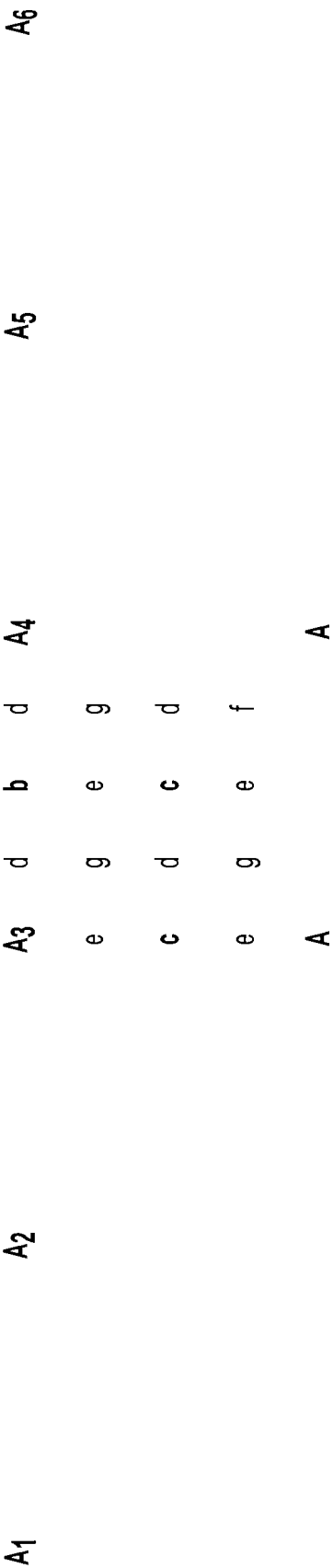


FIG. 13b

A ₁		b ₁		
A ₂		b ₂		
A ₃	d	b ₃	d	A
e	g	e	g	
c	d	c	d	
e	g	e	f	
A ₄		b ₄		A
A ₅		b ₅		
A ₆		b ₆		

FIG. 13c

A	d	b	d	A
e	h	f	h	
b	g	c	g	
e	h	f	i	
A				A

FIG. 14a

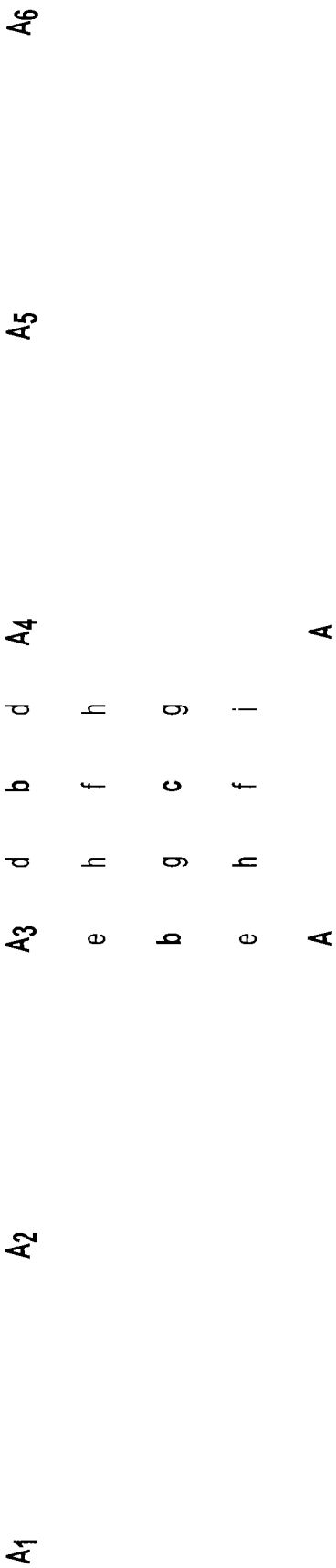


FIG. 14b

A₁

A₂

A ₃	d	b	d	A
e	h	f	h	
b	g	c	g	
e	h	f	i	
A ₄				A

A₅

A₆

FIG. 14c

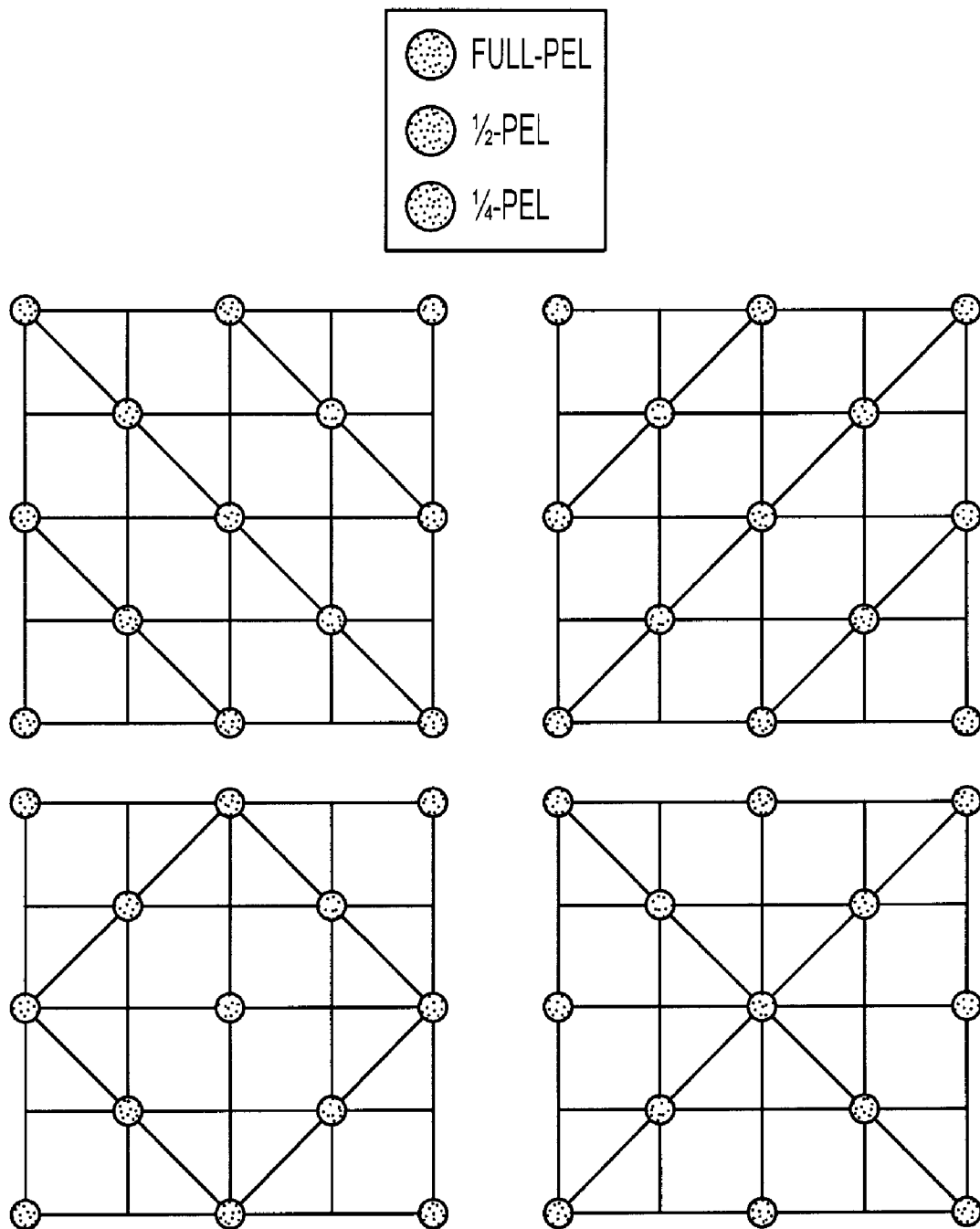


FIG. 15

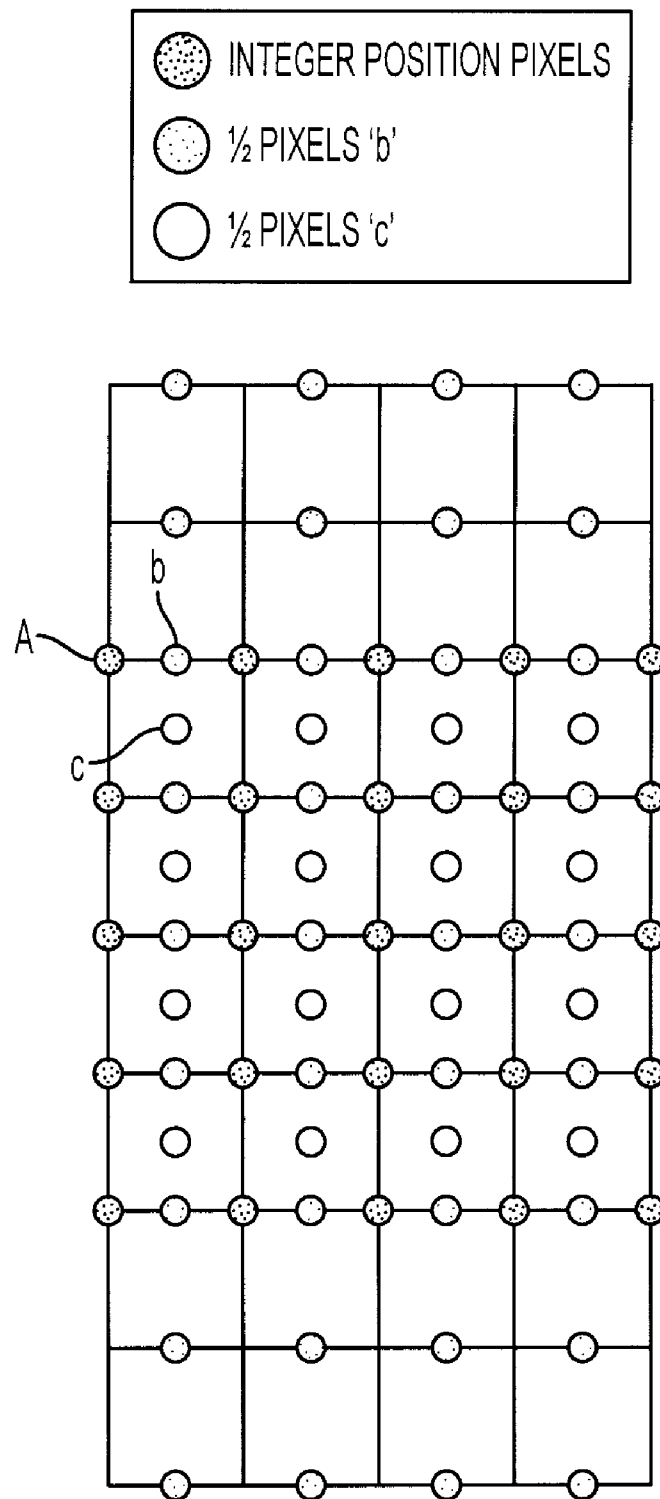


FIG. 16

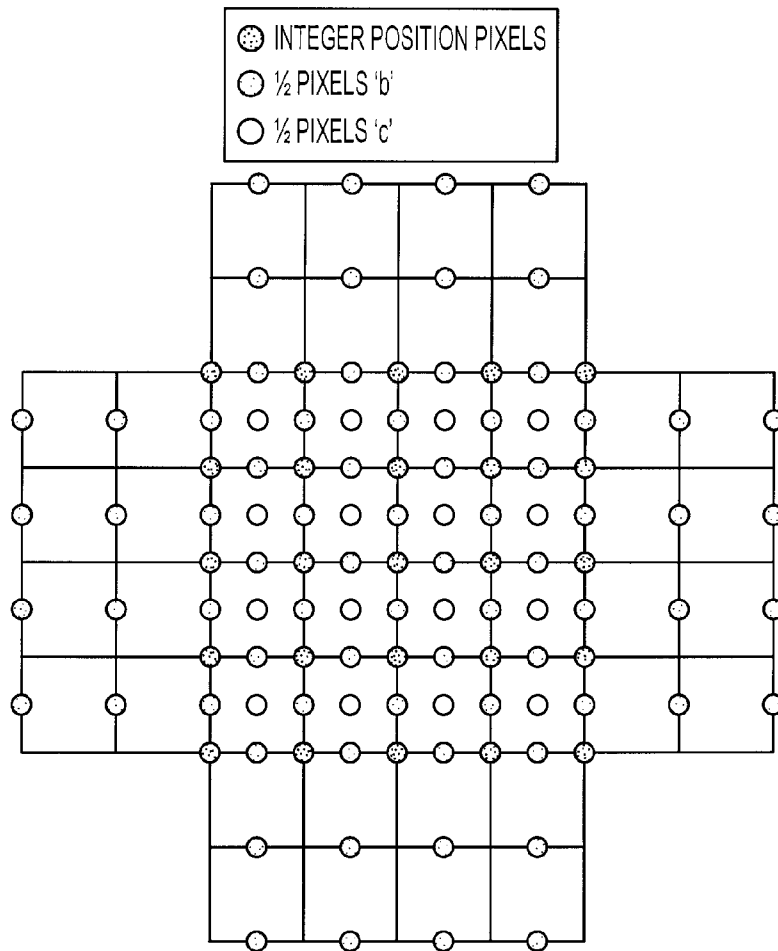


FIG. 17a

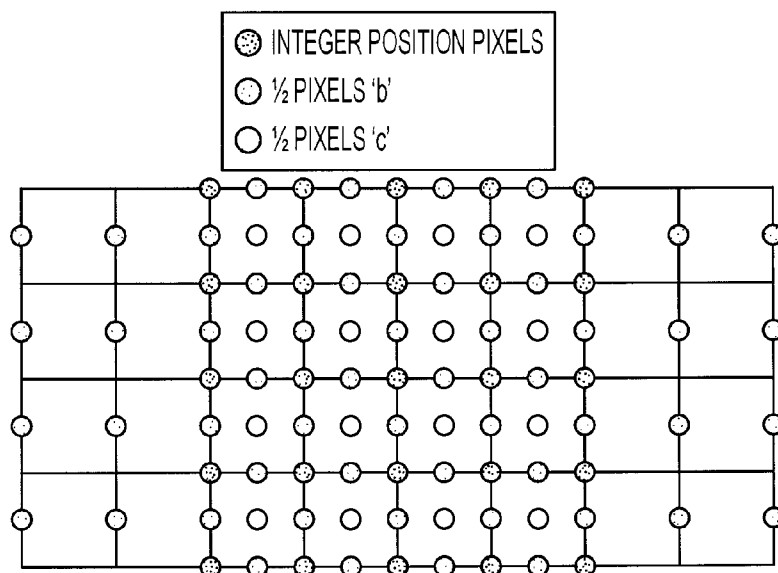


FIG. 17b

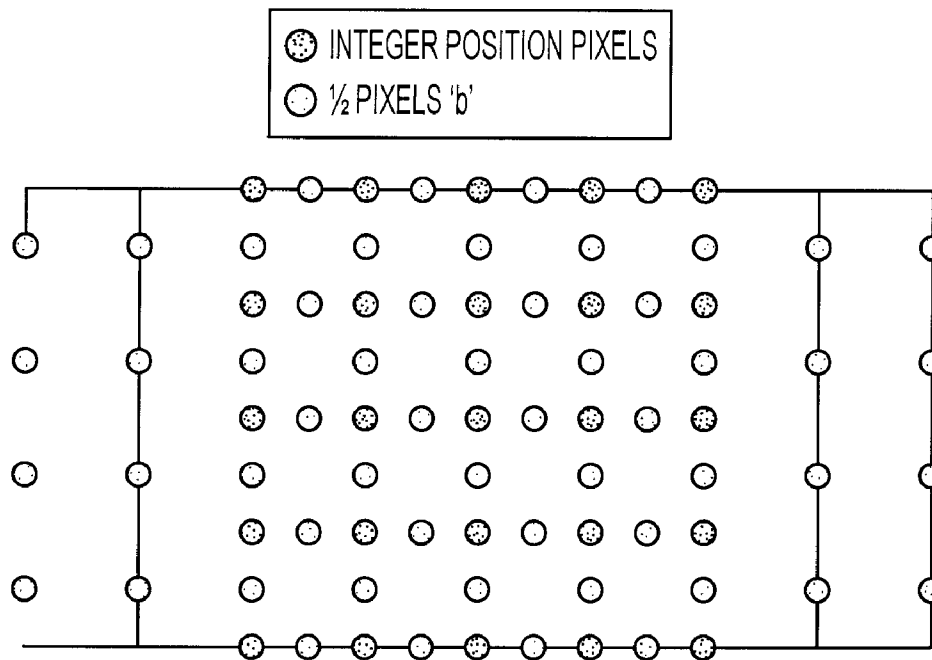


FIG. 18a

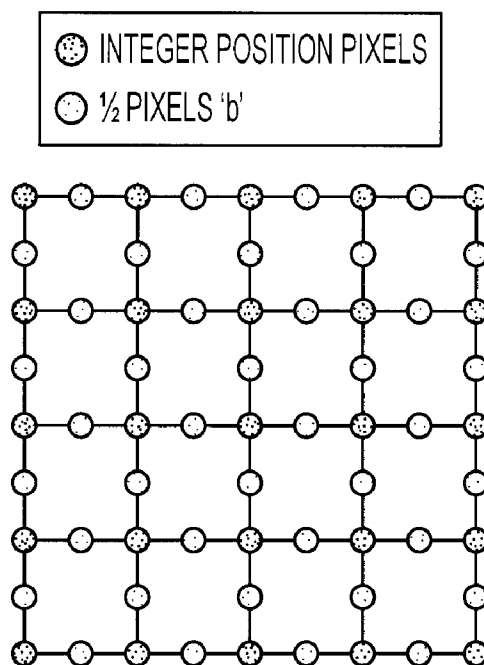


FIG. 18b

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

FIG. 19

A	d	b ¹	d	b ²	d	b ³	d	A
d	e	d	f	d	f	d	e	
b ¹	d	c ¹¹	d	c ¹²	d	c ¹³	d	
d	f	d	g	d	g	d	f	
b ²	d	c ²¹	d	c ²²	d	c ²³	d	
d	f	d	g	d	g	d	e	
b ³	d	c ³¹	d	c ³²	d	c ³³	d	
d	e	d	f	d	f	d	e	
A								

FIG. 20

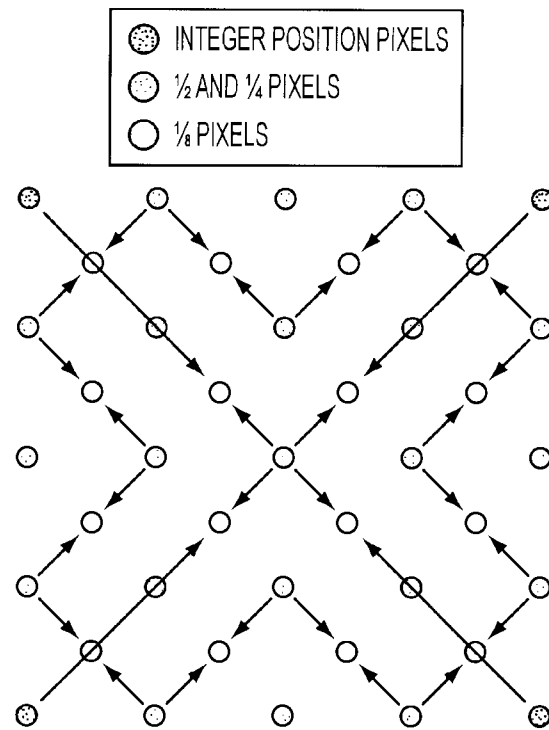


FIG. 21a

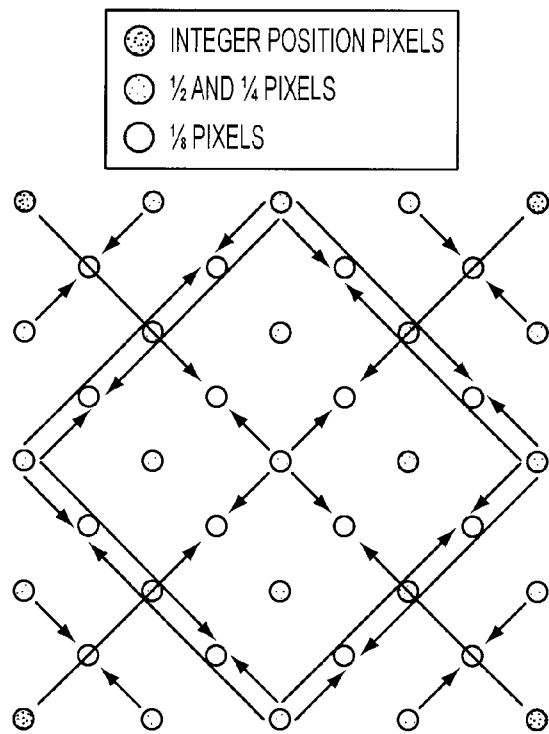


FIG. 21b

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

FIG. 22

METHOD FOR SUB-PIXEL VALUE INTERPOLATION

CROSS REFERENCE TO RELATED APPLICATIONS

This application is a continuation of, and claims priority to, U.S. application Ser. No. 11/090,717, filed Mar. 25, 2005, now allowed, which is a continuation of, and claims priority to, U.S. application Ser. No. 09/954,608, filed Sep. 17, 2001, now U.S. Pat. No. 6,950,469, all of which are incorporated by reference herein in their entirety.

The present invention relates to a method for sub-pixel value interpolation in the encoding and decoding of data. It relates particularly, but not exclusively, to encoding and decoding of digital video.

BACKGROUND OF THE INVENTION

Digital video sequences, like ordinary motion pictures recorded on film, comprise a sequence of still images, the illusion of motion being created by displaying the images one after the other at a relatively fast frame rate, typically 15 to 30 frames per second. Because of the relatively fast frame rate, images in consecutive frames tend to be quite similar and thus contain a considerable amount of redundant information. For example, a typical scene may comprise some stationary elements, such as background scenery, and some moving areas, which may take many different forms, for example the face of a newsreader, moving traffic and so on. Alternatively, the camera recording the scene may itself be moving, in which case all elements of the image have the same kind of motion. In many cases, this means that the overall change between one video frame and the next is rather small. Of course, this depends on the nature of the movement. For example, the faster the movement, the greater the change from one frame to the next. Similarly, if a scene contains a number of moving elements, the change from one frame to the next is likely to be greater than in a scene where only one element is moving.

It should be appreciated that each frame of a raw, that is uncompressed, digital video sequence comprises a very large amount of image information. Each frame of an uncompressed digital video sequence is formed from an array of image pixels. For example, in a commonly used digital video format, known as the Quarter Common Interchange Format (QCIF), a frame comprises an array of 176×144 pixels, in which case each frame has 25,344 pixels. In turn, each pixel is represented by a certain number of bits, which carry information about the luminance and/or colour content of the region of the image corresponding to the pixel. Commonly, a so-called YUV colour model is used to represent the luminance and chrominance content of the image. The luminance, or Y, component represents the intensity (brightness) of the image, while the colour content of the image is represented by two chrominance components, labelled U and V.

Colour models based on a luminance/chrominance representation of image content provide certain advantages compared with colour models that are based on a representation involving primary colours (that is Red, Green and Blue, RGB). The human visual system is more sensitive to intensity variations than it is to colour variations; YUV colour models exploit this property by using a lower spatial resolution for the chrominance components (U, V) than for the luminance component (Y). In this way the amount of information needed to code the colour information in an image can be reduced with an acceptable reduction in image quality.

The lower spatial resolution of the chrominance components is usually attained by sub-sampling. Typically, a block of 16×16 image pixels is represented by one block of 16×16 pixels comprising luminance information and the corresponding chrominance components are each represented by one block of 8×8 pixels representing an area of the image equivalent to that of the 16×16 pixels of the luminance component. The chrominance components are thus spatially sub-sampled by a factor of 2 in the x and y directions. The resulting assembly of one 16×16 pixel luminance block and two 8×8 pixel chrominance blocks is commonly referred to as a YUV macroblock, or macroblock, for short.

A QCIF image comprises 11×9 macroblocks. If the luminance blocks and chrominance blocks are represented with 8 bit resolution (that is by numbers in the range 0 to 255), the total number of bits required per macroblock is $(16 \times 16 \times 8) + 2 \times (8 \times 8 \times 8) = 3072$ bits. The number of bits needed to represent a video frame in QCIF format is thus $99 \times 3072 = 304,128$ bits. This means that the amount of data required to transmit/record/display a video sequence in QCIF format, represented using a YUV colour model, at a rate of 30 frames per second, is more than 9 Mbps (million bits per second). This is an extremely high data rate and is impractical for use in video recording, transmission and display applications because of the very large storage capacity, transmission channel capacity and hardware performance required.

If video data is to be transmitted in real-time over a fixed line network such as an ISDN (Integrated Services Digital Network) or a conventional PSTN (Public Service Telephone Network), the available data transmission bandwidth is typically of the order of 64 kbits/s. In mobile videotelephony, where transmission takes place at least in part over a radio communications link, the available bandwidth can be as low as 20 kbits/s. This means that a significant reduction in the amount of information used to represent video data must be achieved in order to enable transmission of digital video sequences over low bandwidth communication networks. For this reason video compression techniques have been developed which reduce the amount of information transmitted while retaining an acceptable image quality.

Video compression methods are based on reducing the redundant and perceptually irrelevant parts of video sequences. The redundancy in video sequences can be categorised into spatial, temporal and spectral redundancy. 'Spatial redundancy' is the term used to describe the correlation between neighbouring pixels within a frame. The term 'temporal redundancy' expresses the fact that the objects appearing in one frame of a sequence are likely to appear in subsequent frames, while 'spectral redundancy' refers to the correlation between different colour components of the same image.

Sufficiently efficient compression cannot usually be achieved by simply reducing the various forms of redundancy in a given sequence of images. Thus, most current video encoders also reduce the quality of those parts of the video sequence which are subjectively the least important. In addition, the redundancy of the compressed video bit-stream is itself reduced by means of efficient loss-less encoding. Typically, this is achieved using a technique known as 'variable length coding' (VLC).

Modern video compression standards, such as ITU-T recommendations H.261, H.263(+)(++), H.26L and the Motion Picture Experts Group recommendation MPEG-4 make use of 'motion compensated temporal prediction'. This is a form of temporal redundancy reduction in which the content of some (often many) frames in a video sequence is 'predicted'

from other frames in the sequence by tracing the motion of objects or regions of an image between frames.

Compressed images which do not make use of temporal redundancy reduction are usually called INTRA-coded or I-frames, whereas temporally predicted images are called INTER-coded or P-frames. In the case of INTER frames, the predicted (motion-compensated) image is rarely precise enough to represent the image content with sufficient quality, and therefore a spatially compressed prediction error (PE) frame is also associated with each INTER frame. Many video compression schemes can also make use of bi-directionally predicted frames, which are commonly referred to as B-pictures or B-frames. B-pictures are inserted between reference or so-called 'anchor' picture pairs (I or P frames) and are predicted from either one or both of the anchor pictures. B-pictures are not themselves used as anchor pictures, that is no other frames are predicted from them, and therefore, they can be discarded from the video sequence without causing deterioration in the quality of future pictures.

The different types of frame that occur in a typical compressed video sequence are illustrated in FIG. 3 of the accompanying drawings. As can be seen from the figure, the sequence starts with an INTRA or I frame 30. In FIG. 3, arrows 33 denote the 'forward' prediction process by which P-frames (labelled 34) are formed. The bi-directional prediction process by which B-frames (36) are formed is denoted by arrows 31a and 31b, respectively.

A schematic diagram of an example video coding system using motion compensated prediction is shown in FIGS. 1 and 2. FIG. 1 illustrates an encoder 10 employing motion compensation and FIG. 2 illustrates a corresponding decoder 20. The encoder 10 shown in FIG. 1 comprises a Motion Field Estimation block 11, a Motion Field Coding block 12, a Motion Compensated Prediction block 13, a Prediction Error Coding block 14, a Prediction Error Decoding block 15, a Multiplexing block 16, a Frame Memory 17, and an adder 19. The decoder 20 comprises a Motion Compensated Prediction block 21, a Prediction Error Decoding block 22, a Demultiplexing block 23 and a Frame Memory 24.

The operating principle of video coders using motion compensation is to minimise the amount of information in a prediction error frame $E_n(x,y)$, which is the difference between a current frame $I_n(x,y)$ being coded and a prediction frame $P_n(x,y)$. The prediction error frame is thus:

$$E_n(x,y) = I_n(x,y) - P_n(x,y). \quad (1)$$

The prediction frame $P_n(x,y)$ is built using pixel values of a reference frame $R_n(x,y)$, which is generally one of the previously coded and transmitted frames, for example the frame immediately preceding the current frame and is available from the Frame Memory 17 of the encoder 10. More specifically, the prediction frame $P_n(x,y)$ is constructed by finding so-called 'prediction pixels' in the reference frame $R_n(x,y)$ which correspond substantially with pixels in the current frame. Motion information, describing the relationship (e.g. relative location, rotation, scale etc.) between pixels in the current frame and their corresponding prediction pixels in the reference frame is derived and the prediction frame is constructed by moving the prediction pixels according to the motion information. In this way, the prediction frame is constructed as an approximate representation of the current frame, using pixel values in the reference frame. The prediction error frame referred to above therefore represents the difference between the approximate representation of the current frame provided by the prediction frame and the current frame itself. The basic advantage provided by video encoders that use motion compensated prediction arises from the fact

that a comparatively compact description of the current frame can be obtained by representing it in terms of the motion information required to form its prediction together with the associated prediction error information in the prediction error frame.

However, due to the very large number of pixels in a frame, it is generally not efficient to transmit separate motion information for each pixel to the decoder. Instead, in most video coding schemes, the current frame is divided into larger image segments S_k and motion information relating to the segments is transmitted to the decoder. For example, motion information is typically provided for each macroblock of a frame and the same motion information is then used for all pixels within the macroblock. In some video coding standards, such as H.26L, a macroblock can be divided into smaller blocks, each smaller block being provided with its own motion information.

The motion information usually takes the form of motion vectors $[\Delta x(x,y), \Delta y(x,y)]$. The pair of numbers $\Delta x(x,y)$ and $\Delta y(x,y)$ represents the horizontal and vertical displacements of a pixel at location (x,y) in the current frame $I_n(x,y)$ with respect to a pixel in the reference frame $R_n(x,y)$. The motion vectors $[\Delta x(x,y), \Delta y(x,y)]$ are calculated in the Motion Field Estimation block 11 and the set of motion vectors of the current frame $[\Delta x(\cdot), \Delta y(\cdot)]$ is referred to as the motion vector field.

Typically, the location of a macroblock in a current video frame is specified by the (x,y) co-ordinate of its upper left-hand corner. Thus, in a video coding scheme in which motion information is associated with each macroblock of a frame, each motion vector describes the horizontal and vertical displacement $\Delta x(x,y)$ and $\Delta y(x,y)$ of a pixel representing the upper left-hand corner of a macroblock in the current frame $I_n(x,y)$ with respect to a pixel in the upper left-hand corner of a substantially corresponding block of prediction pixels in the reference frame $R_n(x,y)$ (as shown in FIG. 4b).

Motion estimation is a computationally intensive task. Given a reference frame $R_n(x,y)$ and, for example, a square macroblock comprising $N \times N$ pixels in a current frame (as shown in FIG. 4a), the objective of motion estimation is to find an $N \times N$ pixel block in the reference frame that matches the characteristics of the macroblock in the current picture according to some criterion. This criterion can be, for example, a sum of absolute differences (SAD) between the pixels of the macroblock in the current frame and the block of pixels in the reference frame with which it is compared. This process is known generally as 'block matching'. It should be noted that, in general, the geometry of the block to be matched and that in the reference frame do not have to be the same, as real-world objects can undergo scale changes, as well as rotation and warping. However, in current international video coding standards, only a translational motion model is used (see below) and thus fixed rectangular geometry is sufficient.

Ideally, in order to achieve the best chance of finding a match, the whole of the reference frame should be searched. However, this is impractical as it imposes too high a computational burden on the video encoder. Instead, the search region is restricted to region $[-p,p]$ around the original location of the macroblock in the current frame, as shown in FIG. 4c.

In order to reduce the amount of motion information to be transmitted from the encoder 10 to the decoder 20, the motion vector field is coded in the Motion Field Coding block 12 of the encoder 10, by representing it with a motion model. In this process, the motion vectors of image segments are re-expressed using certain predetermined functions or, in other words, the motion vector field is represented with a model.

5

Almost all currently used motion vector field models are additive motion models, complying with the following general formula:

$$\Delta x(x, y) = \sum_{i=0}^{N-1} a_i f_i(x, y) \quad (2)$$

$$\Delta y(x, y) = \sum_{i=0}^{M-1} b_i g_i(x, y) \quad (3)$$

where coefficients a_i and b_i are called motion coefficients. The motion coefficients are transmitted to the decoder **20** (information stream **2** in FIGS. **1** and **2**). Functions f_i and g_i are called motion field basis functions, and are known both to the encoder and decoder. An approximate motion vector field ($\hat{\Delta}x(x, y), \hat{\Delta}y(x, y)$) can be constructed using the coefficients and the basis functions. As the basis functions are known to (that is stored in) both the encoder **10** and the decoder **20**, only the motion coefficients need to be transmitted to the encoder, thus reducing the amount of information required to represent the motion information of the frame.

The simplest motion model is the translational motion model which requires only two coefficients to describe the motion vectors of each segment. The values of motion vectors are given by:

$$\Delta x(x, y) = a_0$$

$$\Delta y(x, y) = b_0 \quad (4)$$

This model is widely used in various international standards (ISO MPEG-1, MPEG-2, MPEG-4, ITU-T Recommendations H.261 and H.263) to describe the motion of 16×16 and 8×8 pixel blocks. Systems which use a translational motion model typically perform motion estimation at full pixel resolution or some integer fraction of full pixel resolution, for example at half or one quarter pixel resolution.

The prediction frame $P_n(x, y)$ is constructed in the Motion Compensated Prediction block **13** in the encoder **10**, and is given by:

$$P_n(x, y) = R_n[x + \hat{\Delta}x(x, y), y + \hat{\Delta}y(x, y)] \quad (5)$$

In the Prediction Error Coding block **14**, the prediction error frame $E_n(x, y)$ is typically compressed by representing it as a finite series (transform) of some 2-dimensional functions. For example, a 2-dimensional Discrete Cosine Transform (DCT) can be used. The transform coefficients are quantised and entropy (for example Huffman) coded before they are transmitted to the decoder (information stream **1** in FIGS. **1** and **2**). Because of the error introduced by quantisation, this operation usually produces some degradation (loss of information) in the prediction error frame $E_n(x, y)$. To compensate for this degradation, the encoder **10** also comprises a Prediction Error Decoding block **15**, where a decoded prediction error frame $\hat{E}_n(x, y)$ is constructed using the transform coefficients. This locally decoded prediction error frame is added to the prediction frame $P_n(x, y)$ in the adder **19** and the resulting decoded current frame $\hat{I}_n(x, y)$ is stored in the Frame Memory **17** for further use as the next reference frame $R_{n+1}(x, y)$.

The information stream **2** carrying information about the motion vectors is combined with information about the prediction error in multiplexer **16** and an information stream **3** containing typically at least those two types of information is sent to the decoder **20**.

6

The operation of a corresponding video decoder **20** will now be described.

The Frame Memory **24** of the decoder **20** stores a previously reconstructed reference frame $R_n(x, y)$. The prediction frame $P_n(x, y)$ is constructed in the Motion Compensated Prediction block **21** of the decoder **20** according to equation 5, using received motion coefficient information and pixel values of the previously reconstructed reference frame $R_n(x, y)$. The transmitted transform coefficients of the prediction error frame $E_n(x, y)$ are used in the Prediction Error Decoding block **22** to construct the decoded prediction error frame $\hat{E}_n(x, y)$. The pixels of the decoded current frame $\hat{I}_n(x, y)$ are then reconstructed by adding the prediction frame $P_n(x, y)$ and the decoded prediction error frame $\hat{E}_n(x, y)$:

$$\hat{I}_n(x, y) = P_n(x, y) + \hat{E}_n(x, y) = R_n[x + \hat{\Delta}x(x, y), y + \hat{\Delta}y(x, y)] + \hat{E}_n(x, y) \quad (6)$$

This decoded current frame may be stored in the Frame Memory **24** as the next reference frame $R_{n+1}(x, y)$.

In the description of motion compensated encoding and decoding of digital video presented above, the motion vector $[\Delta x(x, y), \Delta y(x, y)]$ describing the motion of a macroblock in the current frame with respect to the reference frame $R_n(x, y)$ can point to any of the pixels in the reference frame. This means that motion between frames of a digital video sequence can only be represented at a resolution which is determined by the image pixels in the frame (so-called full pixel resolution). Real motion, however, has arbitrary precision, and thus the system described above can only provide approximate modelling of the motion between successive frames of a digital video sequence. Typically, modelling of motion between video frames with full pixel resolution is not sufficiently accurate to allow efficient minimisation of the prediction error (PE) information associated with each macroblock/frame. Therefore, to enable more accurate modelling of real motion and to help reduce the amount of PE information that must be transmitted from encoder to decoder, many video coding standards, such as H.263(+) and H.26L, allow motion vectors to point 'in between' image pixels. In other words, the motion vectors can have 'sub-pixel' resolution. Allowing motion vectors to have sub-pixel resolution adds to the complexity of the encoding and decoding operations that must be performed, so it is still advantageous to limit the degree of spatial resolution a motion vector may have. Thus, video coding standards, such as those previously mentioned, typically only allow motion vectors to have full-, half- or quarter-pixel resolution.

Motion estimation with sub-pixel resolution is usually performed as a two-stage process, as illustrated in FIG. **5**, for a video coding scheme which allows motion vectors to have full- or half-pixel resolution. In the first step, a motion vector having full-pixel resolution is determined using any appropriate motion estimation scheme, such as the block-matching process described in the foregoing. The resulting motion vector, having full-pixel resolution is shown in FIG. **5**.

In the second stage, the motion vector determined in the first stage is refined to obtain the desired half-pixel resolution. In the example illustrated in FIG. **5**, this is done by forming eight new search blocks of 16×16 pixels, the location of the top-left corner of each block being marked with an X in FIG. **5**. These locations are denoted as $[\Delta x + m/2, \Delta y + n/2]$, where m and n can take the values $-1, 0$ and $+1$, but cannot be zero at the same time. As only the pixel values of original image pixels are known, the values (for example luminance and/or chrominance values) of the sub-pixels residing at half-pixel locations must be estimated for each of the eight new search blocks, using some form of interpolation scheme.

Having interpolated the values of the sub-pixels at half-pixel resolution, each of the eight search blocks is compared with the macroblock whose motion vector is being sought. As in the block matching process performed in order to determine the motion vector with full pixel resolution, the macroblock is compared with each of the eight search blocks according to some criterion, for example a SAD. As a result of the comparisons, a minimum SAD value will generally be obtained. Depending on the nature of the motion in the video sequence, this minimum value may correspond to the location specified by the original motion vector (having full-pixel resolution), or it may correspond to a location having a half-pixel resolution. Thus, it is possible to determine whether a motion vector should point to a full-pixel or sub-pixel location and if sub-pixel resolution is appropriate, to determine the correct sub-pixel resolution motion vector. It should also be appreciated that the scheme just described can be extended to other sub-pixel resolutions (for example, one-quarter-pixel resolution) in an entirely analogous fashion.

In practice, the estimation of a sub-pixel value in the reference frame is performed by interpolating the value of the sub-pixel from surrounding pixel values. In general, interpolation of a sub-pixel value $F(x,y)$ situated at a non-integer location $(x,y)=(n+\Delta x, m+\Delta y)$, can be formulated as a two-dimensional operation, represented mathematically as:

$$F(x, y) = \sum_{k=-K}^{K-1} \sum_{l=-L}^{L-1} f(k+K, l+L) F(n+k, m+l) \quad (7)$$

where $f(k,l)$ are filter coefficients and n and m are obtained by truncating x and y , respectively, to integer values. Typically, the filter coefficients are dependent on the x and y values and the interpolation filters are usually so-called 'separable filters', in which case sub-pixel value $F(x,y)$ can be calculated as follows:

$$F(x, y) = \sum_{k=-K}^{K-1} f(k+K) \sum_{l=-K}^{K-1} f(l+K) F(n+k, m+l) \quad (8)$$

The motion vectors are calculated in the encoder. Once the corresponding motion coefficients are transmitted to the decoder, it is a straightforward matter to interpolate the required sub-pixels using an interpolation method identical to that used in the encoder. In this way, a frame following a reference frame in the Frame Memory 24, can be reconstructed from the reference frame and the motion vectors.

The simplest way of applying sub-pixel value interpolation in a video coder is to interpolate each sub-pixel value every time it is needed. However, this is not an efficient solution in a video encoder, because it is likely that the same sub-pixel value will be required several times and thus calculations to interpolate the same sub-pixel value will be performed multiple times. This results in an unnecessary increase of computational complexity/burden in the encoder.

An alternative approach, which limits the complexity of the encoder, is to pre-calculate and store all sub-pixel values in a memory associated with the encoder. This solution is called interpolation 'before-hand' interpolation hereafter in this document. While limiting complexity, before-hand interpolation has the disadvantage of increasing memory usage by a large margin. For example, if the motion vector accuracy is one quarter pixel in both horizontal and vertical dimensions,

storing pre-calculated sub-pixel values for a complete image results in a memory usage that is 16 times that required to store the original, non-interpolated image. In addition, it involves the calculation of some sub-pixels which might not actually be required in calculating motion vectors in the encoder. Before-hand interpolation is also particularly inefficient in a video decoder, as the majority of pre-calculated sub-pixel values will never be required by the decoder. Thus, it is advantageous not to use pre-calculation in the decoder.

So-called 'on-demand' interpolation can be used to reduce memory requirements in the encoder. For example, if the desired pixel precision is quarter pixel resolution, only sub-pixels at one half unit resolution are interpolated before-hand for the whole frame and stored in the memory. Values of one-quarter pixel resolution sub-pixels are only calculated during the motion estimation/compensation process as and when it is required. In this case memory usage is only 4 times that required to store the original, non-interpolated image.

It should be noted that when before-hand interpolation is used, the interpolation process constitutes only a small fraction of the total encoder computational complexity/burden, since every pixel is interpolated just once. Therefore, in the encoder, the complexity of the interpolation process itself is not very critical when before-hand sub-pixel value interpolation is used. On the other hand, on-demand interpolation poses a much higher computational burden on the encoder, since sub-pixels may be interpolated many times. Hence the complexity of interpolation process, which may be considered in terms of the number of computational operations or operational cycles that must be performed in order to interpolate the sub-pixel values, becomes an important consideration.

In the decoder, the same sub-pixel values are used a few times at most and some are not needed at all. Therefore, in the decoder it is advantageous not to use before-hand interpolation at all, that is, it is advantageous not to pre-calculate any sub-pixel values.

Two interpolation schemes have been developed as part of the work ongoing in the ITU-Telecommunications Standardization Sector, Study Group 16, Video Coding Experts Group (VCEG), Questions 6 and 15. These approaches were proposed for incorporation into ITU-T recommendation H.26L and have been implemented in test models (TML) for the purposes of evaluation and further development. The test model corresponding to Question 15 is referred to as Test Model 5 (TML5), while that resulting from Question 6 is known as Test Model 6 (TML6). The interpolation schemes proposed in both TML5 and TML6 will now be described.

Throughout the description of the sub-pixel value interpolation scheme used in test model TML5, reference will be made to FIG. 12a, which defines a notation for describing pixel and sub-pixel locations specific to TML5. A separate notation, defined in FIG. 13a, will be used in the discussion of the sub-pixel value interpolation scheme used in TML6. A still further notation, illustrated in FIG. 14a, will be used later in the text in connection with the sub-pixel value interpolation method according to the invention. It should be appreciated that the three different notations used in the text are intended to assist in the understanding of each interpolation method and to help distinguish differences between them. However, in all three figures, the letter A is used to denote original image pixels (full pixel resolution). More specifically, the letter A represents the location of pixels in the image data representing a frame of a video sequence, the pixel values of pixels A being either received as current frame $I_n(x,y)$ from a video source, or reconstructed and stored as a reference frame $R_n(x,y)$ in the Frame Memory 17, 24 of the encoder 10 or the

decoder 20. All other letters represent sub-pixel locations, the values of the sub-pixels situated at the sub-pixel locations being obtained by interpolation.

Certain other terms will also be used in a consistent manner throughout the text to identify particular pixel and sub-pixel locations. These are as follows:

The term 'unit horizontal location' is used to describe the location of any sub-pixel that is constructed in a column of the original image data. Sub-pixels c and e in FIGS. 12a and 13a, as well as sub-pixels b and e in FIG. 14a have unit horizontal locations.

The term 'unit vertical location' is used to describe any sub-pixel that is constructed in a row of the original image data. Sub-pixels b and d in FIGS. 12a and 13a as well as sub-pixels b and d in FIG. 14a have unit vertical locations.

By definition, pixels A have unit horizontal and unit vertical locations.

The term 'half horizontal location' is used to describe the location of any sub-pixel that is constructed in a column that lies at half pixel resolution. Sub-pixels b, c, and e shown in FIGS. 12a and 13a fall into this category, as do sub-pixels b, c and f in FIG. 14a. In a similar manner, the term 'half vertical location' is used to describe the location of any sub-pixel that is constructed in a row that lies at half-pixel resolution, such as sub-pixels c and d in FIGS. 12a and 13a, as well as sub-pixels b, c and g in FIG. 14a.

Furthermore, the term 'quarter horizontal location' refers to any sub-pixel that is constructed in a column which lies at quarter-pixel resolution, such as sub-pixels d and e in FIG. 12a, sub-pixels d and g in FIG. 13a and sub-pixels d, g and h in FIG. 14a. Analogously, the term 'quarter vertical location' refers to sub-pixels that are constructed in a row which lies at quarter-pixel resolution. In FIG. 12a, sub-pixels e and f fall into this category, as do sub-pixels e, f and g in FIG. 13a and sub-pixels e, f and h in FIG. 14a.

The definition of each of the terms described above is shown by 'envelopes' drawn on the corresponding figures.

It should further be noted that it is often convenient to denote a particular pixel with a two-dimensional reference. In this case, the appropriate two-dimensional reference can be obtained by examining the intersection of the envelopes in FIGS. 12a, 13a and 14a. Applying this principle, pixel d in FIG. 12a, for example, has a half horizontal and half vertical location and sub-pixel e has a unit horizontal and quarter vertical location. In addition, and for ease of reference, sub-pixels that reside at half unit horizontal and unit vertical locations, unit horizontal and half unit vertical locations as well as half unit horizontal and half unit vertical locations, will be referred to as $\frac{1}{2}$ resolution sub-pixels. Sub-pixels which reside at any quarter unit horizontal and/or quarter unit vertical location will be referred to as $\frac{1}{4}$ resolution sub-pixels.

It should also be noted that in the descriptions of the two test models and in the detailed description of the invention itself, it will be assumed that pixels have a minimum value of 0 and a maximum value of $2^n - 1$ where n is the number of bits reserved for a pixel value. The number of bits is typically 8. After a sub-pixel has been interpolated, if the value of that interpolated sub-pixel exceeds the value of $2^n - 1$, it is restricted to the range of $[0, 2^n - 1]$, i.e. values lower than the minimum allowed value will become the minimum value (0) and values larger than the maximum will become maximum value ($2^n - 1$). This operation is called clipping.

The sub-pixel value interpolation scheme according to TML5 will now be described in detail with reference to FIGS. 12a, 12b and 12c.

1. The value for the sub-pixel at half unit horizontal and unit vertical location, that is $\frac{1}{2}$ resolution sub-pixel b in FIG. 12a, is calculated using a 6-tap filter. The filter interpolates a value for $\frac{1}{2}$ resolution sub-pixel b based upon the values of the 6 pixels (A_1 to A_6) situated in a row at unit horizontal locations and unit vertical locations symmetrically about b, as shown in FIG. 12b, according to the formula $b = (A_1 - 5A_2 + 20A_3 + 20A_4 - 5A_5 + A_6 + 16)/32$. The operator / denotes division with truncation. The result is clipped to lie in the range $[0, 2^n - 1]$.

2. Values for the $\frac{1}{2}$ resolution sub-pixels labelled c are calculated using the same six tap filter as used in step 1 and the six nearest pixels or sub-pixels (A or b) in the vertical direction. Referring now to FIG. 12c, the filter interpolates a value for the $\frac{1}{2}$ resolution sub-pixel c located at unit horizontal and half vertical location based upon the values of the 6 pixels (A_1 to A_6) situated in a column at unit horizontal locations and unit vertical locations symmetrically about c, according to the formula $c = (A_1 - 5A_2 + 20A_3 + 20A_4 - 5A_5 + A_6 + 16)/32$. Similarly, a value for the $\frac{1}{2}$ resolution sub-pixel c at half horizontal and half vertical location is calculated according to $c = (b_1 - 5b_2 + 20b_3 + 20b_4 - 5b_5 + b_6 + 16)/32$. Again, the operator / denotes division with truncation. The values calculated for the c sub-pixels are further clipped to lie in the range $[0, 2^n - 1]$.

At this point in the interpolation process the values of all $\frac{1}{2}$ resolution sub-pixels have been calculated and the process proceeds to the calculation of $\frac{1}{4}$ resolution sub-pixel values.

3. Values for the $\frac{1}{4}$ resolution sub-pixels labelled d are calculated using linear interpolation and the values of the nearest pixels and/or $\frac{1}{2}$ resolution sub-pixels in the horizontal direction. More specifically, values for $\frac{1}{4}$ resolution sub-pixels d located at quarter horizontal and unit vertical locations, are calculated by taking the average of the immediately neighbouring pixel at unit horizontal and unit vertical location (pixel A) and the immediately neighbouring $\frac{1}{2}$ resolution sub-pixel at half horizontal and unit vertical location (sub-pixel b), i.e. according to $d = (A + b)/2$. Values for $\frac{1}{4}$ resolution sub-pixels d located at quarter horizontal and half vertical locations, are calculated by taking the average of the immediately neighbouring $\frac{1}{2}$ resolution sub-pixels c which lie at unit horizontal and half vertical location and half horizontal and half vertical locations respectively, i.e. according to $d = (c_1 + c_2)/2$. Again operator / indicates division with truncation.

4. Values for the $\frac{1}{4}$ resolution sub-pixels labelled e are calculated using linear interpolation and the values of the nearest pixels and/or $\frac{1}{2}$ resolution sub-pixels in the vertical direction. In particular, $\frac{1}{4}$ resolution sub-pixels e at unit horizontal and quarter vertical locations are calculated by taking the average of the immediately neighbouring pixel at unit horizontal and unit vertical location (pixel A) and the immediately neighbouring sub-pixel at unit horizontal and half vertical location (sub-pixel c) according to $e = (A + c)/2$. $\frac{1}{4}$ resolution sub-pixels e_3 at half horizontal and quarter vertical locations are calculated by taking the average of the immediately neighbouring sub-pixel at half horizontal and unit vertical location (sub-pixel b) and the immediately neighbouring sub-pixel at half horizontal and half vertical location (sub-pixel c), according to $e = (b + c)/2$. Furthermore, $\frac{1}{4}$ resolution sub-pixels e at quarter horizontal and quarter vertical locations are calculated by taking the average of the immediately neighbouring sub-pixels at quarter horizontal and unit vertical location and the corresponding sub-pixel at quarter horizontal and half vertical location (sub-pixels d), according to $e = (d_1 + d_2)/2$. Once more, operator / indicates division with truncation.

11

5. The value for $\frac{1}{4}$ resolution sub-pixel f is interpolated by averaging the values of the 4 closest pixels values at unit horizontal and vertical locations, according to $f=(A_1+A_2+A_3+A_4+2)/4$, where pixels A_1, A_2, A_3 and A_4 are the four nearest original pixels.

A disadvantage of TML5 is that the decoder is computationally complex. This results from the fact that TML5 uses an approach in which interpolation of $\frac{1}{4}$ resolution sub-pixel values depends upon the interpolation of $\frac{1}{2}$ resolution sub-pixel values. This means that in order to interpolate the values of the $\frac{1}{4}$ resolution sub-pixels, the values of the $\frac{1}{2}$ resolution sub-pixels from which they are determined must be calculated first. Furthermore, since the values of some of the $\frac{1}{4}$ resolution sub-pixels depend upon the interpolated values obtained for other $\frac{1}{4}$ resolution sub-pixels, truncation of the $\frac{1}{4}$ resolution sub-pixel values has a deleterious effect on the precision of some of the $\frac{1}{4}$ resolution sub-pixel values. Specifically, the $\frac{1}{4}$ resolution sub-pixel values are less precise than they would be if calculated from values that had not been truncated and clipped. Another disadvantage of TML5 is that it is necessary to store the values of the $\frac{1}{2}$ resolution sub-pixels in order to interpolate the $\frac{1}{4}$ resolution sub-pixel values. Therefore, excess memory is required to store a result which is not ultimately required.

The sub-pixel value interpolation scheme according to TML6, referred to herein as direct interpolation, will now be described. In the encoder the interpolation method according to TML6 works like the previously described TML5 interpolation method, except that maximum precision is retained throughout. This is achieved by using intermediate values which are neither rounded nor clipped. A step-by-step description of interpolation method according to TML6 as applied in the encoder is given below with reference to FIGS. 13a, 13b and 13c.

1. The value for the sub-pixel at half unit horizontal and unit vertical location, that is $\frac{1}{2}$ resolution sub-pixel b in FIG. 13a, is obtained by first calculating an intermediate value b using a six tap filter. The filter calculates b based upon the values of the 6 pixels (A_1 to A_6) situated in a row at unit horizontal locations and unit vertical locations symmetrically about b, as shown in FIG. 13b, according to the formula $b=(A_1-5A_2+20A_3+20A_4-5A_5+A_6)$. The final value of b is then calculated as $b=(b+16)/32$ and is clipped to lie in the range $[0, 2^n-1]$. As before, the operator / denotes division with truncation.
2. Values for the $\frac{1}{2}$ resolution sub-pixels labelled c are obtained by first calculating intermediate values c. Referring to FIG. 13c, an intermediate value c for the $\frac{1}{2}$ resolution sub-pixel c located at unit horizontal and half vertical location is calculated based upon the values of the 6 pixels (A_1 to A_6) situated in a column at unit horizontal locations and unit vertical locations symmetrically about c, according to the formula $c=(A_1-5A_2+20A_3+20A_4-5A_5+A_6)$. The final value for the $\frac{1}{2}$ resolution sub-pixel c located at unit horizontal and half vertical location is then calculated according to $c=(c+16)/32$. Similarly, an intermediate value c for the $\frac{1}{2}$ resolution sub-pixel c at half horizontal and half vertical location is calculated according to $c=(b_1-5b_2+20b_3+20b_4-5b_5+b_6)$. A final value for this $\frac{1}{2}$ resolution sub-pixel is then calculated according to $(c+512)/1024$. Again, the operator / denotes division with truncation and the values calculated for $\frac{1}{2}$ resolution sub-pixels c are further clipped to lie in the range $[0, 2^n-1]$.
3. Values for the $\frac{1}{4}$ resolution sub-pixels labelled d are calculated as follows. Values for $\frac{1}{4}$ resolution sub-pixels d located at quarter horizontal and unit vertical locations, are calculated from the value of the immediately neighbouring

12

pixel at unit horizontal and unit vertical location (pixel A) and the intermediate value b calculated in step (1) for the immediately neighbouring $\frac{1}{2}$ resolution sub-pixel at half horizontal and unit vertical location ($\frac{1}{2}$ resolution sub-pixel b), according to $d=(32A+b+32)/64$. Values for $\frac{1}{4}$ resolution sub-pixels d located at quarter horizontal and half vertical locations, are interpolated using the intermediate values c calculated for the immediately neighbouring $\frac{1}{2}$ resolution sub-pixels c which lie at unit horizontal and half vertical location and half horizontal and half vertical locations respectively, according to $d=(32c_1+c_2+1024)/2048$. Again operator / indicates division with truncation and the finally obtained $\frac{1}{4}$ resolution sub-pixel values d are clipped to lie in the range $[0, 2^n-1]$.

4. Values for the $\frac{1}{4}$ resolution sub-pixels labelled e are calculated as follows. Values for $\frac{1}{4}$ resolution sub-pixels e located at unit horizontal and quarter vertical locations are calculated from the value of the immediately neighbouring pixel at unit horizontal and unit vertical location (pixel A) and the intermediate value c calculated in step (2) for the immediately neighbouring $\frac{1}{2}$ resolution sub-pixel at unit horizontal and half vertical location, according to $e=(32A+c+32)/64$. Values for $\frac{1}{4}$ resolution sub-pixels e located at half horizontal and quarter vertical locations are calculated from the intermediate value b calculated in step (1) for the immediately neighbouring $\frac{1}{2}$ resolution sub-pixel at half horizontal and unit vertical location and the intermediate value c calculated in step (2) for the immediately neighbouring $\frac{1}{2}$ resolution sub-pixel at half horizontal and half vertical location, according to $e=(32b+c+1024)/2048$. Once more, operator / indicates division with truncation and the finally obtained $\frac{1}{4}$ resolution sub-pixel values e are clipped to lie in the range $[0, 2^n-1]$.
5. Values for $\frac{1}{4}$ resolution sub-pixels labelled g are computed using the value of the nearest original pixel A and the intermediate values of the three nearest neighbouring $\frac{1}{2}$ resolution sub-pixels, according to $g=(1024A+32b+32c_1+c_2+2048)/4096$. As before, operator / indicates division with truncation and the finally obtained for $\frac{1}{4}$ resolution sub-pixel values g are clipped to lie in the range $[0, 2^n-1]$.
6. The value for $\frac{1}{4}$ resolution sub-pixel f is interpolated by averaging the values of the 4 closest pixels at unit horizontal and vertical locations, according to $f=(A_1+A_2+A_3+A_4+2)/4$, where the locations of pixels A_1, A_2, A_3 and A_4 are the four nearest original pixels.

In the decoder, sub-pixel values can be obtained directly by applying 6-tap filters in horizontal and vertical directions. In the case of $\frac{1}{4}$ sub-pixel resolution, referring to FIG. 13a, the filter coefficients applied to pixels and sub-pixels at unit vertical location are $[0, 0, 64, 0, 0, 0]$ for a set of six pixels A, $[1, -5, 52, 20, -5, 1]$ for a set of six sub-pixels d, $[2, -10, 40, 40, -10, 2]$ for a set of six sub-pixels b, and $[1, -5, 20, 52, -5, 1]$ for a set of six sub-pixels c. These filter coefficients are applied to respective sets of pixels or sub-pixels in the same row as the sub-pixel values being interpolated.

After applying the filters in the horizontal and vertical directions, interpolated value c is normalized according to $c=(c+2048)/4096$ and clipped to lie in the range $[0, 2^n-1]$. When a motion vector points to an integer pixel position in either the horizontal or vertical direction, many zero coefficients are used. In a practical implementation of TML6, different branches are used in the software which are optimised for the different sub-pixel cases so that there are no multiplications by zero coefficients.

It should be noted that in TML6, $\frac{1}{4}$ resolution sub-pixel values are obtained directly using the intermediate values referred to above and are not derived from rounded and

13

clipped values for $\frac{1}{2}$ resolution sub-pixels. Therefore, in obtaining the $\frac{1}{4}$ resolution sub-pixel values, it is not necessary to calculate final values for any of the $\frac{1}{2}$ resolution sub-pixels. Specifically, it is not necessary to carry out the truncation and clipping operations associated with the calculation of final values for the $\frac{1}{2}$ resolution sub-pixels. Neither is it necessary to have stored final values for $\frac{1}{2}$ resolution sub-pixels for use in the calculation of the $\frac{1}{4}$ resolution sub-pixel values. Therefore TML6 is computationally less complex than TML5, as fewer truncating and clipping operations are required. However, a disadvantage of TML6 is that high precision arithmetic is required both in the encoder and in the decoder. High precision interpolation requires more silicon area in ASICs and requires more computations in some CPUs. Furthermore, implementation of direct interpolation as specified in TML6 in an on-demand fashion has a high memory requirement. This is an important factor, particularly in embedded devices.

In view of the previously presented discussion, it should be appreciated that due to the different requirements of the video encoder and decoder with regard to sub-pixel interpolation, there exists a significant problem in developing a method of sub-pixel value interpolation capable of providing satisfactory performance in both the encoder and decoder. Furthermore, neither of the current test models (TML5, TML6) described in the foregoing can provide a solution that is optimum for application in both encoder and decoder.

SUMMARY OF THE INVENTION

According to a first aspect of the invention there is provided method of interpolation in video coding in which an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, is interpolated to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $\frac{1}{2}^x$, where x is a positive integer having a maximum value N, the method comprising:

- a) when values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations are required, interpolating such values directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) when values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations are required, interpolating such values directly using a choice of a first weighted sum of values for sub-pixels residing at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a); and
- c) when a value for a sub-pixel situated at a $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical location is required, interpolating such a value by taking a weighted average of the value of a first sub-pixel or pixel situated at a $\frac{1}{2}^{N-m}$ unit horizontal and $\frac{1}{2}^{N-n}$ unit vertical location and the value of a second sub-pixel or pixel located at a $\frac{1}{2}^{N-p}$ unit horizontal and $\frac{1}{2}^{N-q}$ unit vertical location, variables m, n, p and q taking integer values in the range 1 to N such that the first and second sub-pixels or pixels are located diagonally with respect to the sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical location.

Preferably a first and a second weight are used in the weighted average referred to in (c), the relative magnitudes of the weights being inversely proportional to the (straight-line

14

diagonal) proximity of the first and the second sub-pixel or pixel to the sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical location.

In a situation where the first and the second sub-pixel or pixel are symmetrically located with respect to (equidistant from) the sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical location, the first and second weights may have equal values.

The first weighted sum of values for sub-pixels residing at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations in step b) may be used when a sub-pixel at $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^N$ unit vertical location is required.

The second weighted sum of values for sub-pixels residing at unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations in step b) may be used when a sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location is required.

In one embodiment, when values for sub-pixels at $\frac{1}{2}^N$ unit horizontal and unit vertical locations, and $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations are required, such values are interpolated by taking the average of the values of a first pixel or sub-pixel located at a vertical location corresponding to that of the sub-pixel being calculated and unit horizontal location and a second pixel or sub-pixel located at a vertical location corresponding to that of the sub-pixel being calculated and $\frac{1}{2}^{N-1}$ unit horizontal location.

When values for sub-pixels at unit horizontal and $\frac{1}{2}^N$ unit vertical locations, and $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations are required, they may be interpolated by taking the average of the values of a first pixel or sub-pixel located at a horizontal location corresponding to that of the sub-pixel being calculated and unit vertical location and a second pixel or sub-pixel located at a horizontal location corresponding to that of the sub-pixel being calculated and $\frac{1}{2}^{N-1}$ unit vertical location.

Values for sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations may be interpolated by taking the average of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

Values for sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations may be interpolated by taking the average of values of a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

Values for half of the sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations may be interpolated by taking the average of a first pair of values of a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location and values for the other half of the sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations are interpolated by taking the average of a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

Values for sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations are alternately interpolated for one such sub-pixel by taking the average of a first pair of values of a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location and values and for a neighbouring such sub-pixel by taking the average of a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

The sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations may be alternately interpolated in a horizontal direction.

15

The sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations may be alternately interpolated in a horizontal direction.

When values for some sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations are required, such values may be interpolated by taking the average of a plurality of nearest neighbouring pixels.

At least one of step a) and step b) interpolating sub-pixel values directly using weighted sums may involve the calculation of an intermediate value for the sub-pixel values having a dynamic range greater than the specified dynamic range.

The intermediate value for a sub-pixel having $\frac{1}{2}^{N-1}$ sub-pixel resolution may be used the calculation of a sub-pixel value having $\frac{1}{2}^N$ sub-pixel resolution.

According to a second aspect of the invention, there is provided a method of interpolation in video coding in which an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, is interpolated to generate values for sub-pixels at fractional horizontal and vertical locations, the method comprising:

- a) when values for sub-pixels at half unit horizontal and unit vertical locations, and unit horizontal and half unit vertical locations are required, interpolating such values directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) when values for sub-pixels at half unit horizontal and half unit vertical locations are required, interpolating such values directly using a weighted sum of values for sub-pixels residing at half unit horizontal and unit vertical locations calculated according to step (a); and
- c) when values for sub-pixels at quarter unit horizontal and quarter unit vertical locations are required, interpolating such values by taking the average of at least one pair of a first pair of values of a sub-pixel located at a half unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and half unit vertical location and a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a half unit horizontal and half unit vertical location.

According to a third aspect of the invention, there is provided a method of interpolation in video coding in which an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, is interpolated to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $\frac{1}{2}^x$ where x is a positive integer having a maximum value N, the method comprising:

- a) when values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations are required, interpolating such values directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) when a value for a sub-pixel at a sub-pixel horizontal and sub-pixel vertical location is required, interpolating such a value directly using a choice of a first weighted sum of values for sub-pixels located at a vertical location corresponding to that of the sub-pixel being calculated and a second weighted sum of values for sub-pixels located at a horizontal location corresponding to that of the sub-pixel being calculated.

16

The sub-pixels used in the first weighted sum may be sub-pixels residing at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations and the first weighted sum may be used to interpolate a value for a sub-pixel at $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^N$ unit vertical location.

The sub-pixels used in the second weighted sum may be sub-pixels residing at unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations and the second weighted sum may be used to interpolate a value for a sub-pixel at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

When values for sub-pixels at $\frac{1}{2}^N$ unit horizontal and $\frac{1}{2}^N$ unit vertical locations are required, they may be interpolated by taking the average of at least one pair of a first pair of values of a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location and a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a $\frac{1}{2}^{N-1}$ unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical location.

In the foregoing, N may be equal an integer selected from a list consisting of the values 2, 3, and 4.

Sub-pixels at quarter unit horizontal location are to be interpreted as being sub-pixels having as their left-hand nearest neighbour a pixel at unit horizontal location and as their right-hand nearest neighbour a sub-pixel at half unit horizontal location as well as sub-pixels having as their left-hand nearest neighbour a sub-pixel at half unit horizontal location and as their right-hand nearest neighbour a pixel at unit horizontal location. Correspondingly, sub-pixels at quarter unit vertical location are to be interpreted as being sub-pixels having as their upper nearest neighbour a pixel at unit vertical location and as their lower nearest neighbour a sub-pixel at half unit vertical location as well as sub-pixels having as their upper nearest neighbour a sub-pixel at half unit vertical location and as their lower nearest neighbour a pixel at unit vertical location.

The term dynamic range, refers to the range of values which the sub-pixel values and the weighted sums can take.

Preferably changing the dynamic range, whether by extending it or reducing it, means changing the number of bits which are used to represent the dynamic range.

In an embodiment of the invention, the method is applied to an image that is sub-divided into a number of image blocks. Preferably each image block comprises four corners, each corner being defined by a pixel located at a unit horizontal and unit vertical location. Preferably the method is applied to each image block as the block becomes available for sub-pixel value interpolation. Alternatively, sub-pixel value interpolation according to the method of the invention is performed once all image blocks of an image have become available for sub-pixel value interpolation.

Preferably the method is used in video encoding. Preferably the method is used in video decoding.

In one embodiment of the invention, when used in encoding, the method is carried out as before-hand interpolation, in which values for all sub-pixels at half unit locations and values for all sub-pixels at quarter unit locations are calculated and stored before being subsequently used in the determination of a prediction frame during motion predictive coding. In alternative embodiments, the method is carried out as a combination of before-hand and on-demand interpolation. In this case, a certain proportion or category of sub-pixel values is calculated and stored before being used in the determination of a prediction frame and certain other sub-pixel values are calculated only when required during motion predictive coding.

17

Preferably, when the method is used in decoding, sub-pixels are only interpolated when their need is indicated by a motion vector.

According to a fourth aspect of the invention, there is provided a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the video coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $1/2^x$, where x is a positive integer having a maximum value N, the interpolator being adapted to:

- a) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $1/2^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and $1/2^{N-1}$ unit vertical locations directly using a choice of a first weighted sum of values for sub-pixels residing at $1/2^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $1/2^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a); and
- c) interpolate a value for a sub-pixel situated at a $1/2^N$ unit horizontal and $1/2^N$ unit vertical location by taking a weighted average of the value of a first sub-pixel or pixel situated at a $1/2^{N-m}$ unit horizontal and $1/2^{N-n}$ unit vertical location and the value of a second sub-pixel or pixel located at a $1/2^{N-p}$ unit horizontal and $1/2^{N-q}$ unit vertical location, variables m, n, p and q taking integer values in the range 1 to N such that the first and second sub-pixels or pixels are located diagonally with respect to the sub-pixel at $1/2^N$ unit horizontal and $1/2^N$ unit vertical location.

The video coder may comprise a video encoder. It may comprise a video decoder. There may be a codec comprising both the video encoder and the video decoder.

According to a fifth aspect of the invention, there is provided a communications terminal comprising a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the video coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $1/2^x$, where x is a positive integer having a maximum value N, the interpolator being adapted to:

- a) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $1/2^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and $1/2^{N-1}$ unit vertical locations directly using a choice of a first weighted sum of values for sub-pixels residing at $1/2^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $1/2^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a); and
- c) interpolate a value for a sub-pixel situated at a $1/2^N$ unit horizontal and $1/2^N$ unit vertical location by taking a weighted average of the value of a first sub-pixel or pixel situated at a $1/2^{N-m}$ unit horizontal and $1/2^{N-n}$ unit vertical

18

location and the value of a second sub-pixel or pixel located at a $1/2^{N-p}$ unit horizontal and $1/2^{N-q}$ unit vertical location, variables m, n, p and q taking integer values in the range 1 to N such that the first and second sub-pixels or pixels are located diagonally with respect to the sub-pixel at $1/2^N$ unit horizontal and $1/2^N$ unit vertical location.

The communications terminal may comprise a video encoder. It may comprise a video decoder. Preferably, it comprises a video codec comprising a video encoder and a video decoder.

Preferably the communications terminal comprising a user interface, a processor and at least one of a transmitting block and a receiving block, and a video coder according to at least one of the third and fourth aspects of the invention. Preferably the processor controls the operation of the transmitting block and/or the receiving block and the video coder.

According to a sixth aspect of the invention, there is provided a telecommunications system comprising a communications terminal and a network, the telecommunications network and the communications terminal being connected by a communications link over which coded video can be transmitted, the communications terminal comprising a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the video coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $1/2^x$, where x is a positive integer having a maximum value N, the interpolator being adapted to:

- a) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $1/2^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and $1/2^{N-1}$ unit vertical locations directly using a choice of a first weighted sum of values for sub-pixels residing at $1/2^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $1/2^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a); and
- c) interpolate a value for a sub-pixel situated at a $1/2^N$ unit horizontal and $1/2^N$ unit vertical location by taking a weighted average of the value of a first sub-pixel or pixel situated at a $1/2^{N-m}$ unit horizontal and $1/2^{N-n}$ unit vertical location and the value of a second sub-pixel or pixel located at a $1/2^{N-p}$ unit horizontal and $1/2^{N-q}$ unit vertical location, variables m, n, p and q taking integer values in the range 1 to N such that the first and second sub-pixels or pixels are located diagonally with respect to the sub-pixel at $1/2^N$ unit horizontal and $1/2^N$ unit vertical location.

Preferably the telecommunications system is a mobile telecommunications system comprising a mobile communications terminal and a wireless network, the connection between the mobile communications terminal and the wireless network being formed by a radio link. Preferably the network enables the communications terminal to communicate with other communications terminals connected to the network over communications links between the other communications terminals and the network.

According to a seventh aspect of the invention, there is provided a telecommunications system comprising a communications terminal and a network, the telecommunications network and the communications terminal being connected

by a communications link over which coded video can be transmitted, the network comprising a video coder for coding for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the video coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the fractional horizontal and vertical locations being defined according to $1/2^x$, where x is a positive integer having a maximum value N, the interpolator being adapted to:

- a) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $1/2^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and $1/2^{N-1}$ unit vertical locations directly using a choice of a first weighted sum of values for sub-pixels residing at $1/2^{N-1}$ unit horizontal and unit vertical locations and a second weighted sum of values for sub-pixels residing at unit horizontal and $1/2^{N-1}$ unit vertical locations, the first and second weighted sums of values being calculated according to step (a); and
- c) interpolate a value for a sub-pixel situated at a $1/2^N$ unit horizontal and $1/2^N$ unit vertical location by taking a weighted average of the value of a first sub-pixel or pixel situated at a $1/2^{N-m}$ unit horizontal and $1/2^{N-n}$ unit vertical location and the value of a second sub-pixel or pixel located at a $1/2^{N-p}$ unit horizontal and $1/2^{N-q}$ unit vertical location, variables m, n, p and q taking integer values in the range 1 to N such that the first and second sub-pixels or pixels are located diagonally with respect to the sub-pixel at $1/2^N$ unit horizontal and $1/2^N$ unit vertical location.

According to an eighth aspect of the invention there is provided a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the resolution of the sub-pixels being determined by a positive integer N, the interpolator being adapted to:

- a) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $1/2^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate a value for a sub-pixel at a sub-pixel horizontal and sub-pixel vertical location is required directly using a choice of a first weighted sum of values for sub-pixels located at a vertical location corresponding to that of the sub-pixel being calculated and a second weighted sum of values for sub-pixels located at a horizontal location corresponding to that of the sub-pixel being calculated.

The interpolator may be further adapted to form the first weighted sum using the values of sub-pixels residing at $1/2^{N-1}$ unit horizontal and unit vertical locations and to use the first weighted sum to interpolate a value for a sub-pixel at $1/2^{N-1}$ unit horizontal and $1/2^N$ unit vertical location.

The interpolator may be further adapted to form the second weighted sum using the values of sub-pixels residing at unit horizontal and $1/2^{N-1}$ unit vertical locations and to use the second weighted sum to interpolate a value for a sub-pixel at $1/2^N$ unit horizontal and $1/2^{N-1}$ unit vertical location.

The interpolator may be further adapted to interpolate values for sub-pixels at $1/2^N$ unit horizontal and $1/2^N$ unit vertical

locations by taking the average of at least one pair of a first pair of values of a sub-pixel located at a $1/2^{N-1}$ unit horizontal and unit vertical location, and a sub-pixel located at a unit horizontal and $1/2^{N-1}$ unit vertical location and a second pair of values of a pixel located at a unit horizontal and unit vertical location, and a sub-pixel located at a $1/2^{N-1}$ unit horizontal and $1/2^{N-1}$ unit vertical location.

According to a ninth aspect of the invention there is provided a communications terminal comprising a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the resolution of the sub-pixels being determined by a positive integer N, the interpolator being adapted to:

- a) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $1/2^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate a value for a sub-pixel at a sub-pixel horizontal and sub-pixel vertical location is required directly using a choice of a first weighted sum of values for sub-pixels located at a vertical location corresponding to that of the sub-pixel being calculated and a second weighted sum of values for sub-pixels located at a horizontal location corresponding to that of the sub-pixel being calculated.

According to a tenth aspect of the invention there is provided a telecommunications system comprising a communications terminal and a network, the telecommunications network and the communications terminal being connected by a communications link over which coded video can be transmitted, the communications terminal comprising a video coder for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal and vertical locations, the resolution of the sub-pixels being determined by a positive integer N, the interpolator being adapted to:

- a) interpolate values for sub-pixels at $1/2^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $1/2^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate a value for a sub-pixel at a sub-pixel horizontal and sub-pixel vertical location is required directly using a choice of a first weighted sum of values for sub-pixels located at a vertical location corresponding to that of the sub-pixel being calculated and a second weighted sum of values for sub-pixels located at a horizontal location corresponding to that of the sub-pixel being calculated.

According to an eleventh aspect of the invention there is provided a telecommunications system comprising a communications terminal and a network, the telecommunications network and the communications terminal being connected by a communications link over which coded video can be transmitted, the network comprising a video coder for coding for coding an image comprising pixels arranged in rows and columns and represented by values having a specified dynamic range, the pixels in the rows residing at unit horizontal locations and the pixels in the columns residing at unit vertical locations, the coder comprising an interpolator adapted to generate values for sub-pixels at fractional horizontal

zontal and vertical locations, the resolution of the sub-pixels being determined by a positive integer N, the interpolator being adapted to:

- a) interpolate values for sub-pixels at $\frac{1}{2}^{N-1}$ unit horizontal and unit vertical locations, and unit horizontal and $\frac{1}{2}^{N-1}$ unit vertical locations directly using weighted sums of pixels residing at unit horizontal and unit vertical locations;
- b) interpolate a value for a sub-pixel at a sub-pixel horizontal and sub-pixel vertical location is required directly using a choice of a first weighted sum of values for sub-pixels located at a vertical location corresponding to that of the sub-pixel being calculated and a second weighted sum of values for sub-pixels located at a horizontal location corresponding to that of the sub-pixel being calculated.

According to the twelfth aspect of the invention, the disclosed embodiments are directed to a method for sub-pixel value interpolation to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being arranged in rows and columns, the pixel and sub-pixel locations being representable mathematically within the rectangular bounded region using the co-ordinate notation $K/2^N$, $L/2^N$, K and L being positive integers having respective values between zero and 2^N , N being a positive integer greater than one and representing a particular degree of sub-pixel value interpolation, the method comprising:

- (a) interpolating a sub-pixel value for a sub-pixel having co-ordinates with odd values of both K and L, according to a predetermined choice of a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates $1/2$, $1/2$, and a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with even values of both K and L, including zero, situated within a quadrant of the rectangular bounded region defined by corner pixels having co-ordinates $1/2$, $1/2$ and the nearest neighbouring pixel;
- (b) interpolating sub-pixel values for sub-pixels having co-ordinates with K equal to an even value and L equal zero and sub-pixels having co-ordinates with K equal to zero and L equal to an even value, used in the interpolation of the sub-pixels having co-ordinates with odd values of both K and L, using weighted sums of the values of pixels located in rows and columns respectively; and
- (c) interpolating sub-pixel values for sub-pixels having co-ordinates with even values of both K and L, used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of both K and L, using a predetermined choice of either a weighted sum of the values of sub-pixels having co-ordinates with K equal to an even value and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, or a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent bounded rectangular regions.

According to the thirteenth aspect of the invention, the disclosed embodiments are directed to a method for quarter resolution sub-pixel value interpolation to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being arranged in rows and columns, the pixel and sub-pixel locations being

representable mathematically within the rectangular bounded region using the co-ordinate notation $K/4$, $L/4$, K and L being positive integers having respective values between zero and 4, the method comprising:

- (a) interpolating a sub-pixel value for a sub-pixel having co-ordinates with both K and L equal to 1 or 3, according to a predetermined choice of a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates $2/4$, $2/4$, and a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with both K and L equal to zero, 2 or 4, situated within a quadrant of the rectangular bounded region defined by corner pixels having co-ordinates $2/4$, $2/4$ and the nearest neighbouring pixel;
- (b) interpolating sub-pixel values for sub-pixels having co-ordinates with K equal to 2 and L equal zero and sub-pixels having co-ordinates with K equal to zero and L equal to 2, used in the interpolation of the sub-pixels having co-ordinates with both K and L equal to 1 or 3, using weighted sums of the values of pixels located in rows and columns respectively; and
- (c) interpolating a sub-pixel value for the sub-pixel having co-ordinates with both K and L equal to 2, used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with both K and L equal to 1 or 3, using a predetermined choice of either a weighted sum of the values of the sub-pixel having co-ordinates with K equal to 2 and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, or a weighted sum of the value of the sub-pixel having co-ordinates with K equal to zero and L equal to 2 and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions.

According to the fourteenth aspect of the invention, the disclosed embodiments are directed to a method for eighth resolution sub-pixel value interpolation to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being arranged in rows and columns, the pixel and sub-pixel locations being representable mathematically with the rectangular bounded region using the co-ordinate notation $K/8$, $L/8$, K and L being positive integers having respective values between zero and 8, the method comprising:

- (a) interpolating a sub-pixel value for a sub-pixel having co-ordinates with both K and L equal to 1, 3, 5 or 7, according to a predetermined choice of a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates $4/8$, $4/8$, and a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with both K and L equal to zero, 2, 4, 6 or 8, situated within a quadrant of the rectangular bounded region defined by corner pixels having co-ordinates $4/8$, $4/8$ and the nearest neighbouring pixel;
- (b) interpolating sub-pixel values for sub-pixels having co-ordinates with K equal to 2, 4 or 6 and L equal zero and sub-pixels having co-ordinates with K equal to zero and L equal to 2, 4 or 6, used in the interpolation of the sub-pixels having co-ordinates with both K and L equal to 1, 3, 5 or 7, using weighted sums of the values of pixels located in rows and columns respectively; and
- (c) interpolating sub-pixel values for sub-pixels having co-ordinates with both K and L equal to 2, 4 or 6, used in the interpolation of sub-pixel values for the sub-pixels

23

having co-ordinates with both K and L equal to 1, 3, 5 or 7, using a predetermined choice of either a weighted sum of the values of sub-pixels having co-ordinates with K equal to 2, 4 or 6 and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions or a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to 2, 4 or 6 and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions.

According to the fifteenth aspect of the invention, the disclosed embodiments are directed to an apparatus for sub-pixel value interpolation, the apparatus being operable to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being arranged in rows and columns, the pixel and sub-pixel locations being representable mathematically within the rectangular bounded region using the co-ordinate notation $K/2^N$, $L/2^N$, K and L being positive integers having respective values between zero and 2^N , N being a positive integer greater than one and representing a particular degree of sub-pixel value interpolation, the apparatus comprising:

- (a) circuitry operable to interpolate a sub-pixel value for a sub-pixel having co-ordinates with odd values of both K and L, according to a predetermined choice of a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates 1/2, 1/2, and a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with even values of both K and L, including zero, situated within a quadrant of the rectangular bounded region defined by corner pixels having co-ordinates 1/2, 1/2 and the nearest neighbouring pixel;
- (b) circuitry operable to interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an even value and L equal zero and sub-pixels having co-ordinates with K equal to zero and L equal to an even value, used in the interpolation of the sub-pixels having co-ordinates with odd values of both K and L, using weighted sums of the values of pixels located in rows and columns respectively; and
- (c) circuitry operable to interpolate sub-pixel values for sub-pixels having co-ordinates with even values of both K and L, used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of both K and L, using a predetermined choice of either a weighted sum of the values of sub-pixels having co-ordinates with K equal to an even value and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, or a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent bounded rectangular regions.

In one embodiment the method includes comprising using a first and a second weight in the weighted average chosen to interpolate the sub-pixel value for a sub-pixel having co-ordinates with odd values of both K and L, wherein when the chosen weighted average is that involving the value of a nearest neighbouring pixel and the value of the sub-pixel situated at co-ordinates 1/2, 1/2, selecting the relative magnitudes of the first and second weights to be inversely proportional to the respective straight-line diagonal distances of the nearest-neighbouring pixel and the sub-pixel situated at co-

24

ordinates 1/2, 1/2 to the sub-pixel having co-ordinates with odd values of both K and L whose value is being interpolated.

In one embodiment the method includes comprising using a first and a second weight in the weighted average chosen to interpolate the sub-pixel value for a sub-pixel having co-ordinates with odd values of both K and L, wherein when the chosen weighted average is that involving the values of a pair of diagonally-opposed sub-pixels having co-ordinates with even values of both K and L, selecting the relative magnitudes of the first and second weights to be inversely proportional to the respective straight-line diagonal distances of the sub-pixels having co-ordinates with even values of both K and L to the sub-pixel having co-ordinates with odd values of both K and L whose value is being interpolated.

In one embodiment the method includes comprising using first and second weights with equal values when the nearest-neighbouring pixel and sub-pixel situated at co-ordinates 1/2, 1/2 are equidistant from the sub-pixel having co-ordinates with odd values of both K and L whose value is being interpolated.

In one embodiment the method includes comprising using first and second weights with equal values when the sub-pixels having co-ordinates with even values of both K and L are equidistant from the sub-pixel having co-ordinates with odd values of both K and L whose value is being interpolated.

In one embodiment the method includes interpolating sub-pixel values for sub-pixels having co-ordinates with even values of both K and L, used in interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of both K and L, using a weighted sum of the values of sub-pixels having co-ordinates with K equal to an even value and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions when the value of a sub-pixel having co-ordinates with K equal to an even value and L equal to an odd value is also required.

In one embodiment the method includes comprising interpolating sub-pixel values for sub-pixels having co-ordinates with even values of both K and L, used in interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of both K and L, using a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions when the value of a sub-pixel having co-ordinates with K equal to an odd value and L equal to an even value is also required.

In one embodiment the method includes comprising interpolating sub-pixel values for sub-pixels having co-ordinates with K equal to an odd value O and L equal to an even value E, not including zero or 2^N , by taking an average of the value of a first sub-pixel having co-ordinates with K equal to the even value O-1 and L equal to E and a second sub-pixel having co-ordinates with K equal to the even value O+1 and L equal to E.

In one embodiment the method includes comprising interpolating a sub-pixel value for a sub-pixel having co-ordinates with K equal to 1 and L equal to zero, by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to zero, and a sub-pixel having co-ordinates with K equal to 2 and L equal to zero.

In one embodiment the method includes comprising interpolating a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N-1 and L equal to zero, by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to zero, and a sub-pixel having co-ordinates with K equal to 2^N-2 and L equal to zero.

25

In one embodiment the method includes comprising interpolating a sub-pixel value for a sub-pixel having co-ordinates with K equal to 1 and L equal to 2^N , by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to 2 and L equal to 2^N .

In one embodiment the method includes comprising interpolating a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N-1 and L equal to 2^N , by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to 2^N-2 and L equal to 2^N .

In one embodiment the method includes comprising interpolating sub-pixel values for sub-pixels having co-ordinates with K equal to an even value E, not including zero or 2^N , and L equal to an odd value O, by taking an average of the value of a first sub-pixel having co-ordinates with K equal to E and L equal to the even value O-1 and a second sub-pixel having co-ordinates with K equal to E and L equal to the even value O+1.

In one embodiment the method includes comprising interpolating a sub-pixel value for a sub-pixel having co-ordinates with K equal to zero and L equal to 1, by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to zero, and a sub-pixel having co-ordinates with K equal to zero and L equal to 2.

In one embodiment the method includes comprising interpolating a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N and L equal to 1, by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to zero, and a sub-pixel having co-ordinates with K equal to zero and L equal to 2.

In one embodiment the method includes comprising interpolating a sub-pixel value for a sub-pixel having co-ordinates with K equal to 0 and L equal to 2^N-1 , by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to zero and L equal to 2^N-2 .

In one embodiment the method includes comprising interpolating a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N and L equal to 2^N-1 , by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to 2^N and L equal to 2^N-2 .

In one embodiment the method includes comprising interpolating a sub-pixel value for the sub-pixel having co-ordinates with K equal to 2^N-1 and L equal to 2^N-1 , by taking a weighted average of the values of the four corner pixels that define the rectangular bounded region.

In one embodiment the method includes comprising selecting a value for N from a list consisting of the values 2, 3, and 4.

In one embodiment the apparatus is operable to use a first and a second weight in the weighted average chosen to interpolate the sub-pixel value for a sub-pixel having co-ordinates with odd values of both K and L, wherein when the chosen weighted average is that involving the value of a nearest neighbouring pixel and the value of the sub-pixel situated at co-ordinates 1/2, 1/2, the apparatus is operable to select the relative magnitudes of the first and second weights to be inversely proportional to the respective straight-line diagonal distances of the nearest-neighbouring pixel and the sub-pixel situated at co-ordinates 1/2, 1/2 to the sub-pixel having co-ordinates with odd values of both K and L whose value is being interpolated.

In one embodiment the apparatus is operable to use a first and a second weight in the weighted average chosen to inter-

26

polate the sub-pixel value for a sub-pixel having co-ordinates with odd values of both K and L, wherein when the chosen weighted average is that involving the values of a pair of diagonally-opposed sub-pixels having co-ordinates with even values of both K and L, the apparatus is operable to select the relative magnitudes of the first and second weights to be inversely proportional to the respective straight-line diagonal distances of the sub-pixels having co-ordinates with even values of both K and L to the sub-pixel having co-ordinates with odd values of both K and L whose value is being interpolated.

In one embodiment the apparatus is operable to use first and second weights with equal values when the nearest-neighbouring pixel and sub-pixel situated at co-ordinates 1/2, 1/2 are equidistant from the sub-pixel having co-ordinates with odd values of both K and L whose value is being interpolated.

In one embodiment the apparatus is operable to use first and second weights with equal values when the sub-pixels having co-ordinates with even values of both K and L are equidistant from the sub-pixel having co-ordinates with odd values of both K and L whose value is being interpolated.

In one embodiment the apparatus is operable to interpolate sub-pixel values for sub-pixels having co-ordinates with even values of both K and L, used in interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of both K and L, using a weighted sum of the values of sub-pixels having co-ordinates with K equal to an even value and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions when the value of a sub-pixel having co-ordinates with K equal to an even value and L equal to an odd value is also required.

In one embodiment the apparatus is operable to interpolate sub-pixel values for sub-pixels having co-ordinates with even values of both K and L, used in interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of both K and L, using a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions when the value of a sub-pixel having co-ordinates with K equal to an odd value and L equal to an even value is also required.

In one embodiment the apparatus is operable to interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an odd value O and L equal to an even value E, not including zero or 2^N , by taking an average of the value of a first sub-pixel having co-ordinates with K equal to the even value O-1 and L equal to E and a second sub-pixel having co-ordinates with K equal to the even value O+1 and L equal to E.

In one embodiment the apparatus is operable to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 1 and L equal to zero, by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to zero, and a sub-pixel having co-ordinates with K equal to 2 and L equal to zero.

In one embodiment the apparatus is operable to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N-1 and L equal to zero, by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to zero, and a sub-pixel having co-ordinates with K equal to 2^N-2 and L equal to zero.

In one embodiment the apparatus is operable to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 1 and L equal to 2^N , by taking an average of the value

27

of a pixel having co-ordinates with K equal to zero and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to 2 and L equal to 2^N .

In one embodiment the apparatus is operable to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N-1 and L equal to 2^N , by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to 2^N-2 and L equal to 2^N .

In one embodiment the apparatus is operable to interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an even value E, not including zero or 2^N , and L equal to an odd value O, by taking an average of the value of a first sub-pixel having co-ordinates with K equal to E and L equal to the even value O-1 and a second sub-pixel having co-ordinates with K equal to E and L equal to the even value O+1.

In one embodiment the apparatus is operable to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to zero and L equal to 1, by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to zero, and a sub-pixel having co-ordinates with K equal to zero and L equal to 2.

In one embodiment the apparatus is operable to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N and L equal to 1, by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to zero, and a sub-pixel having co-ordinates with K equal to zero and L equal to 2.

In one embodiment the apparatus is operable to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 0 and L equal to 2^N-1 , by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to zero and L equal to 2^N-2 .

In one embodiment the apparatus is operable to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N and L equal to 2^N-1 , by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to 2^N and L equal to 2^N-2 .

In one embodiment the apparatus is operable to interpolate a sub-pixel value for the sub-pixel having co-ordinates with K equal to 2^N-1 and L equal to 2^N-1 , by taking a weighted average of the values of the four corner pixels that define the rectangular bounded region.

In one embodiment the apparatus includes N is set equal to 2.

In one embodiment the apparatus includes N is set equal to 3.

In one embodiment the apparatus includes a video encoder comprising an apparatus for sub-pixel value interpolation.

In one embodiment the apparatus includes a still image encoder comprising an apparatus for sub-pixel value interpolation.

In one embodiment the apparatus includes a video decoder comprising an apparatus for sub-pixel value interpolation.

In one embodiment the apparatus includes a still image decoder comprising an apparatus for sub-pixel value interpolation.

In one embodiment the apparatus includes a codec comprising a video encoder and a video decoder.

In one embodiment the apparatus includes a codec comprising a still image encoder and a still image decoder.

In one embodiment the apparatus includes a communications terminal comprising a video encoder.

In one embodiment the apparatus includes a communications terminal comprising a still image encoder.

28

In one embodiment the apparatus includes a communications terminal comprising a video decoder.

In one embodiment the apparatus includes a communications terminal comprising a still image decoder.

BRIEF DESCRIPTION OF THE FIGURES

An embodiment of the invention will now be described by way of example only with reference to the accompanying drawings in which:

FIG. 1 shows a video encoder according to the prior art;

FIG. 2 shows a video decoder according to the prior art;

FIG. 3 shows the types of frames used in video encoding;

FIGS. 4a, 4b, and 4c show steps in block-matching;

FIG. 5 illustrates the process of motion estimation to sub-pixel resolution;

FIG. 6 shows a terminal device comprising video encoding and decoding equipment in which the method of the invention may be implemented;

FIG. 7 shows a video encoder according an embodiment of the present invention;

FIG. 8 shows a video decoder according to an embodiment of the present invention;

FIGS. 9 and 10 have not been used and any such figures should be disregarded;

FIG. 11 shows a schematic diagram of a mobile telecommunications network according to an embodiment of the present invention;

FIG. 12a shows a notation for describing pixel and sub-pixel locations specific to TML5;

FIG. 12b shows interpolation of a half resolution sub-pixels;

FIG. 12c shows interpolation of a half resolution sub-pixels;

FIG. 13a shows a notation for describing pixel and sub-pixel locations specific to TML6;

FIG. 13b shows interpolation of a half resolution sub-pixels;

FIG. 13c shows interpolation of a half resolution sub-pixels;

FIG. 14a shows a notation for describing pixel and sub-pixel locations specific to the invention;

FIG. 14b shows interpolation of a half resolution sub-pixels according to the invention;

FIG. 14c shows interpolation of a half resolution sub-pixels according to the invention;

FIG. 15 shows possible choices of diagonal interpolation for sub-pixels;

FIG. 16 shows the half resolution sub-pixel values required to calculate other half resolution sub-pixel values;

FIG. 17a shows the half resolution sub-pixel values that must be calculated in order to interpolate values for quarter resolution sub-pixels in an image block using the interpolation method of TML5;

FIG. 17b shows the half resolution sub-pixel values that must be calculated in order to interpolate values for quarter resolution sub-pixels in an image block using the interpolation method according to the invention;

FIG. 18a shows the numbers of half resolution sub-pixels that must be calculated in order to obtain values for quarter resolution sub-pixels within an image block using the sub-pixel value interpolation method according to TML5;

FIG. 18b shows the numbers of half resolution sub-pixels that must be calculated in order to obtain values for quarter resolution sub-pixels within an image block using the sub-pixel value interpolation method according to the invention;

FIG. 19 shows a numbering scheme for each of the 15 sub-pixel positions;

FIG. 20 shows nomenclature used to describe pixels, half resolution sub-pixels, quarter resolution sub-pixels and eighth resolution sub-pixels

FIG. 21a shows the diagonal direction to be used in the interpolation of each eighth resolution sub-pixel in an embodiment of the invention;

FIG. 21b shows the diagonal direction to be used in the interpolation of each eighth resolution sub-pixel in another embodiment of the invention; and

FIG. 22 shows nomenclature used to describe eighth resolution sub-pixels within an image.

DETAILED DESCRIPTION

FIGS. 1 to 5, 12a, 12b, 12c, 13a, 13b, and 13c have been described in the foregoing.

FIG. 6 presents a terminal device comprising video encoding and decoding equipment which may be adapted to operate in accordance with the present invention. More precisely, the figure illustrates a multimedia terminal 60 implemented according to ITU-T recommendation H.324. The terminal can be regarded as a multimedia transceiver device. It includes elements that capture, encode and multiplex multimedia data streams for transmission via a communications network, as well as elements that receive, de-multiplex, decode and display received multimedia content. ITU-T recommendation H.324 defines the overall operation of the terminal and refers to other recommendations that govern the operation of its various constituent parts. This kind of multimedia terminal can be used in real-time applications such as conversational videotelephony, or non real-time applications such as the retrieval/streaming of video clips, for example from a multimedia content server in the Internet.

In the context of the present invention, it should be appreciated that the H.324 terminal shown in FIG. 6 is only one of a number of alternative multimedia terminal implementations suited to application of the inventive method. It should also be noted that a number of alternatives exist relating to the location and implementation of the terminal equipment. As illustrated in FIG. 6, the multimedia terminal may be located in communications equipment connected to a fixed line telephone network such as an analogue PSTN (Public Switched Telephone Network). In this case the multimedia terminal is equipped with a modem 71, compliant with ITU-T recommendations V.8, V.34 and optionally V.8bis. Alternatively, the multimedia terminal may be connected to an external modem. The modem enables conversion of the multiplexed digital data and control signals produced by the multimedia terminal into an analogue form suitable for transmission over the PSTN. It further enables the multimedia terminal to receive data and control signals in analogue form from the PSTN and to convert them into a digital data stream that can be demultiplexed and processed in an appropriate manner by the terminal.

An H.324 multimedia terminal may also be implemented in such a way that it can be connected directly to a digital fixed line network, such as an ISDN (Integrated Services Digital Network). In this case the modem 71 is replaced with an ISDN user-network interface. In FIG. 6, this ISDN user-network interface is represented by alternative block 72.

H.324 multimedia terminals may also be adapted for use in mobile communication applications. If used with a wireless communication link, the modem 71 can be replaced with any appropriate wireless interface, as represented by alternative block 73 in FIG. 6. For example, an H.324/M multimedia

terminal can include a radio transceiver enabling connection to the current 2nd generation GSM mobile telephone network, or the proposed 3rd generation UMTS (Universal Mobile Telephone System).

It should be noted that in multimedia terminals designed for two-way communication, that is for transmission and reception of video data, it is advantageous to provide both a video encoder and video decoder implemented according to the present invention. Such an encoder and decoder pair is often implemented as a single combined functional unit, referred to as a 'codec'.

Because a video encoder according to the invention performs motion compensated video encoding to sub-pixel resolution using a specific interpolation scheme and a particular combination of before-hand and on-demand sub-pixel value interpolation, it is generally necessary for a video decoder of a receiving terminal to be implemented in a manner compatible with the encoder of the transmitting terminal which formed the compressed video data stream. Failure to ensure the motion compensation and the accuracy of reconstructed video frames.

A typical H.324 multimedia terminal will now be described in further detail with reference to FIG. 6.

The multimedia terminal 60 includes a variety of elements referred to as 'terminal equipment'. This includes video, audio and telematic devices, denoted generically by reference numbers 61, 62 and 63, respectively. The video equipment 61 may include, for example, a video camera for capturing video images, a monitor for displaying received video content and optional video processing equipment. The audio equipment 62 typically includes a microphone, for example for capturing spoken messages, and a loudspeaker for reproducing received audio content. The audio equipment may also include additional audio processing units. The telematic equipment 63, may include a data terminal, keyboard, electronic whiteboard or a still image transceiver, such as a fax unit.

The video equipment 61 is coupled to a video codec 65. The video codec 65 comprises a video encoder and a corresponding video decoder both implemented according to the invention. Such an encoder and a decoder will be described in the following. The video codec 65 is responsible for encoding captured video data in an appropriate form for further transmission over a communications link and decoding compressed video content received from the communications network. In the example illustrated in FIG. 6, the video codec is implemented according to ITU-T recommendation H.263, with appropriate modifications to implement the sub-pixel value interpolation method according to the invention in both the encoder and the decoder of the video codec.

Similarly, the terminal's audio equipment is coupled to an audio codec, denoted in FIG. 6 by reference number 66. Like the video codec, the audio codec comprises an encoder/decoder pair. It converts audio data captured by the terminal's audio equipment into a form suitable for transmission over the communications link and transforms encoded audio data received from the network back into a form suitable for reproduction, for example on the terminal's loudspeaker. The output of the audio codec is passed to a delay block 67. This compensates for the delays introduced by the video coding process and thus ensures synchronisation of audio and video content.

The system control block 64 of the multimedia terminal controls end-to-network signalling using an appropriate control protocol (signalling block 68) to establish a common mode of operation between a transmitting and a receiving terminal. The signalling block 68 exchanges information

31

about the encoding and decoding capabilities of the transmitting and receiving terminals and can be used to enable the various coding modes of the video encoder. The system control block **64** also controls the use of data encryption. Information regarding the type of encryption to be used in data transmission is passed from encryption block **69** to the multiplexer/de-multiplexer (MUX/DMUX unit) **70**.

During data transmission from the multimedia terminal, the MUX/DMUX unit **70** combines encoded and synchronised video and audio streams with data input from the telematic equipment **63** and possible control data, to form a single bit-stream. Information concerning the type of data encryption (if any) to be applied to the bit-stream, provided by encryption block **69**, is used to select an encryption mode. Correspondingly, when a multiplexed and possibly encrypted multimedia bit-stream is being received, MUX/DMUX unit **70** is responsible for decrypting the bit-stream, dividing it into its constituent multimedia components and passing those components to the appropriate codec(s) and/or terminal equipment for decoding and reproduction.

It should be noted that the functional elements of the multimedia terminal, video encoder, decoder and video codec according to the invention can be implemented as software or dedicated hardware, or a combination of the two. The video encoding and decoding methods according to the invention are particularly suited for implementation in the form of a computer program comprising machine-readable instructions for performing the functional steps of the invention. As such, the encoder and decoder according to the invention may be implemented as software code stored on a storage medium and executed in a computer, such as a personal desktop computer, in order to provide that computer with video encoding and/or decoding functionality.

If the multimedia terminal **60** is a mobile terminal, that is if it is equipped with a radio transceiver **73**, it will be understood by those skilled in the art that it may also comprise additional elements. In one embodiment it comprises a user interface having a display and a keyboard, which enables operation of the multimedia terminal **60** by a user, together with necessary functional blocks including a central processing unit, such as a microprocessor, which controls the blocks responsible for different functions of the multimedia terminal, a random access memory RAM, a read only memory ROM, and a digital camera. The microprocessor's operating instructions, that is program code corresponding to the basic functions of the multimedia terminal **60**, is stored in the read-only memory ROM and can be executed as required by the microprocessor, for example under control of the user. In accordance with the program code, the microprocessor uses the radio transceiver **73** to form a connection with a mobile communication network, enabling the multimedia terminal **60** to transmit information to and receive information from the mobile communication network over a radio path.

The microprocessor monitors the state of the user interface and controls the digital camera. In response to a user command, the microprocessor instructs the camera to record digital images into the RAM. Once an image is captured, or alternatively during the capturing process, the microprocessor segments the image into image segments (for example macroblocks) and uses the encoder to perform motion compensated encoding for the segments in order to generate a compressed image sequence as explained in the foregoing description. A user may command the multimedia terminal **60** to display the captured images on its display or to send the compressed image sequence using the radio transceiver **73** to another multimedia terminal, a video telephone connected to a fixed line network (PSTN) or some other telecommunica-

32

tions device. In a preferred embodiment, transmission of image data is started as soon as the first segment is encoded so that the recipient can start a corresponding decoding process with a minimum delay.

FIG. **11** is a schematic diagram of a mobile telecommunications network according to an embodiment of the invention. Multimedia terminals MS are in communication with base stations BTS by means of a radio link. The base stations BTS are further connected, through a so-called Abis interface, to a base station controller BSC, which controls and manages several base stations.

The entity formed by a number of base stations BTS (typically, by a few tens of base stations) and a single base station controller BSC, controlling the base stations, is called a base station subsystem BSS. Particularly, the base station controller BSC manages radio communication channels and handovers. The base station controller BSC is also connected, through a so-called A interface, to a mobile services switching centre MSC, which co-ordinates the formation of connections to and from mobile stations. A further connection is made, through the mobile service switching centre MSC, to outside the mobile communications network. Outside the mobile communications network there may further reside other network(s) connected to the mobile communications network by gateway(s) GTW, for example the Internet or a Public Switched Telephone Network (PSTN). In such an external network, or within the telecommunications network, there may be located video decoding or encoding stations, such as computers PC. In an embodiment of the invention, the mobile telecommunications network comprises a video server VSRVR to provide video data to a MS subscribing to such a service. The video data is compressed using the motion compensated video compression method as described in the foregoing. The video server may function as a gateway to an online video source or it may comprise previously recorded video clips. Typical videotelephony applications may involve, for example, two mobile stations or one mobile station MS and a videotelephone connected to the PSTN, a PC connected to the Internet or a H.261 compatible terminal connected either to the Internet or to the PSTN.

FIG. **7** shows a video encoder **700** according to an embodiment the invention. FIG. **8** shows a video decoder **800** according to an embodiment the invention.

The encoder **700** comprises an input **701** for receiving a video signal from a camera or other video source (not shown). It further comprises a DCT transformer **705**, a quantiser **706**, an inverse quantiser **709**, an inverse DCT transformer **710**, combiners **712** and **716**, a before-hand sub-pixel interpolation block **730**, a frame store **740** and an on-demand sub-pixel interpolation block **750**, implemented in combination with motion estimation block **760**. The encoder also comprises a motion field coding block **770** and a motion compensated prediction block **780**. Switches **702** and **714** are operated co-operatively by a control manager **720** to switch the encoder between an INTRA-mode of video encoding and an INTER-mode of video encoding. The encoder **700** also comprises a multiplexer unit (MUX/DMUX) **790** to form a single bit-stream from the various types of information produced by the encoder **700** for further transmission to a remote receiving terminal, or for example for storage on a mass storage medium such as a computer hard drive (not shown).

It should be noted that the presence and implementations of before-hand sub-pixel interpolation block **730** and on-demand sub-pixel value interpolation block **750** in the encoder architecture depend on the way in which the sub-pixel interpolation method according to the invention is applied. In embodiments of the invention in which before-hand sub-pixel

33

value interpolation is not performed, encoder 700 does not comprise before-hand sub-pixel value interpolation block 730. In other embodiments of the invention, only before-hand sub-pixel interpolation is performed and thus the encoder does not include on-demand sub-pixel value interpolation block 750. In embodiments in which both before-hand and on-demand sub-pixel value interpolation are performed, both blocks 730 and 750 are present in the encoder 700.

Operation of the encoder 700 according to the invention will now be described in detail. In the description, it will be assumed that each frame of uncompressed video, received from the video source at the input 701, is received and processed on a macroblock-by-macroblock basis, preferably in raster-scan order. It will further be assumed that when the encoding of a new video sequence starts, the first frame of the sequence is encoded in INTRA-mode. Subsequently, the encoder is programmed to code each frame in INTER-format, unless one of the following conditions is met: 1) it is judged that the current frame being coded is so dissimilar from the reference frame used in its prediction that excessive prediction error information is produced; 2) a predefined INTRA frame repetition interval has expired; or 3) feedback is received from a receiving terminal indicating a request for a frame to be coded in INTRA format.

The occurrence of condition 1) is detected by monitoring the output of the combiner 716. The combiner 716 forms a difference between the current macroblock of the frame being coded and its prediction, produced in the motion compensated prediction block 780. If a measure of this difference (for example a sum of absolute differences of pixel values) exceeds a predetermined threshold, the combiner 716 informs the control manager 720 via a control line 717 and the control manager 720 operates the switches 702 and 714 so as to switch the encoder 700 into INTRA coding mode. Occurrence of condition 2) is monitored by means of a timer or frame counter implemented in the control manager 720, in such a way that if the timer expires, or the frame counter reaches a predetermined number of frames, the control manager 720 operates the switches 702 and 714 to switch the encoder into INTRA coding mode. Condition 3) is triggered if the control manager 720 receives a feedback signal from, for example, a receiving terminal, via control line 718 indicating that an INTRA frame refresh is required by the receiving terminal. Such a condition might arise, for example, if a previously transmitted frame were badly corrupted by interference during its transmission, rendering it impossible to decode at the receiver. In this situation, the receiver would issue a request for the next frame to be encoded in INTRA format, thus re-initializing the coding sequence.

It will further be assumed that the encoder and decoder are implemented in such a way as to allow the determination of motion vectors with a spatial resolution of up to quarter-pixel resolution. As will be seen in the following, finer levels of resolution are also possible.

Operation of the encoder 700 in INTRA coding mode will now be described. In INTRA-mode, the control manager 720 operates the switch 702 to accept video input from input line 719. The video signal input is received macroblock by macroblock from input 701 via the input line 719 and each macroblock of original image pixels is transformed into DCT coefficients by the DCT transformer 705. The DCT coefficients are then passed to the quantiser 706, where they are quantised using a quantisation parameter QP. Selection of the quantisation parameter QP is controlled by the control manager 720 via control line 722. Each DCT transformed and quantised macroblock that makes up the INTRA coded image information 723 of the frame is passed from the quantiser 706

34

to the MUX/DMUX 790. The MUX/DMUX 790 combines the INTRA coded image information with possible control information (for example header data, quantisation parameter information, error correction data etc.) to form a single bit-stream of coded image information 725. Variable length coding (VLC) is used to reduce redundancy of the compressed video bit-stream, as is known to those skilled in the art.

A locally decoded picture is formed in the encoder 700 by passing the data output by the quantiser 706 through inverse quantiser 709 and applying an inverse DCT transform 710 to the inverse-quantised data. The resulting data is then input to the combiner 712. In INTRA mode, switch 714 is set so that the input to the combiner 712 via the switch 714 is set to zero. In this way, the operation performed by the combiner 712 is equivalent to passing the decoded image data formed by the inverse quantiser 709 and the inverse DCT transform 710 unaltered.

In embodiments of the invention in which before-hand sub-pixel value interpolation is performed, the output from combiner 712 is applied to before-hand sub-pixel interpolation block 730. The input to the before-hand sub-pixel value interpolation block 730 takes the form of decoded image blocks. In the before-hand sub-pixel value interpolation block 730, each decoded macroblock is subjected to sub-pixel interpolation in such a way that a predetermined sub-set of sub-pixel resolution sub-pixel values is calculated according to the interpolation method of the invention and is stored together with the decoded pixel values in frame store 740.

In embodiments in which before-hand sub-pixel interpolation is not performed, before-hand sub-pixel interpolation block is not present in the encoder architecture and the output from combiner 712, comprising decoded image blocks, is applied directly to frame store 740.

As subsequent macroblocks of the current frame are received and undergo the previously described coding and decoding steps in blocks 705, 706, 709, 710, 712, a decoded version of the INTRA frame is built up in the frame store 740. When the last macroblock of the current frame has been INTRA coded and subsequently decoded, the frame store 740 contains a completely decoded frame, available for use as a prediction reference frame in coding a subsequently received video frame in INTER format. In embodiments of the invention in which before-hand sub-pixel value interpolation is performed, the reference frame held in frame store 740 is at least partially interpolated to sub-pixel resolution.

Operation of the encoder 700 in INTER coding mode will now be described. In INTER coding mode, the control manager 720 operates switch 702 to receive its input from line 721, which comprises the output of the combiner 716. The combiner 716 forms prediction error information representing the difference between the current macroblock of the frame being coded and its prediction, produced in the motion compensated prediction block 780. The prediction error information is DCT transformed in block 705 and quantised in block 706 to form a macroblock of DCT transformed and quantised prediction error information. Each macroblock of DCT transformed and quantised prediction error information is passed from the quantiser 706 to the MUX/DMUX unit 790. The MUX/DMUX unit 790 combines the prediction error information 723 with motion coefficients 724 (described in the following) and control information (for example header data, quantisation parameter information, error correction data etc.) to form a single bit-stream of coded image information, 725.

Locally decoded prediction error information for the each macroblock of the INTER coded frame is then formed in the encoder 700 by passing the encoded prediction error infor-

35

mation 723 output by the quantiser 706 through the inverse quantiser 709 and applying an inverse DCT transform in block 710. The resulting locally decoded macroblock of prediction error information is then input to combiner 712. In INTER-mode, switch 714 is set so that the combiner 712 also receives motion predicted macroblocks for the current INTER frame, produced in the motion compensated prediction block 780. The combiner 712 combines these two pieces of information to produce reconstructed image blocks for the current INTER frame.

As described above when considering INTRA coded frames, in embodiments of the invention in which before-hand sub-pixel value interpolation is performed, the output from combiner 712 is applied to the before-hand sub-pixel interpolation block 730. Thus, the input to the before-hand sub-pixel value interpolation block 730 in INTER coding mode also takes the form of decoded image blocks. In the before-hand sub-pixel value interpolation block 730, each decoded macroblock is subjected to sub-pixel interpolation in such a way that a predetermined sub-set of sub-pixel values is calculated according to the interpolation method of the invention and is stored together with the decoded pixel values in frame store 740. In embodiments in which before-hand sub-pixel interpolation is not performed, before-hand sub-pixel interpolation block is not present in the encoder architecture and the output from combiner 712, comprising decoded image blocks, is applied directly to frame store 740.

As subsequent macroblocks of the video signal are received from the video source and undergo the previously described coding and decoding steps in blocks 705, 706, 709, 710, 712, a decoded version of the INTER frame is built up in the frame store 740. When the last macroblock of the frame has been INTER coded and subsequently decoded, the frame store 740 contains a completely decoded frame, available for use as a prediction reference frame in encoding a subsequently received video frame in INTER format. In embodiments of the invention in which before-hand sub-pixel value interpolation is performed, the reference frame held in frame store 740 is at least partially interpolated to sub-pixel resolution.

Formation of a prediction for a macroblock of the current frame will now be described.

Any frame encoded in INTER format requires a reference frame for motion compensated prediction. This means, inter alia, that when encoding a video sequence, the first frame to be encoded, whether it is the first frame in the sequence, or some other frame, must be encoded in INTRA format. This, in turn, means that when the video encoder 700 is switched into INTER coding mode by control manager 720, a complete reference frame, formed by locally decoding a previously encoded frame, is already available in the frame store 740 of the encoder. In general, the reference frame is formed by locally decoding either an INTRA coded frame or an INTER coded frame.

The first step in forming a prediction for a macroblock of the current frame is performed by motion estimation block 760. The motion estimation block 760 receives the current macroblock of the frame being coded via line 727 and performs a block matching operation in order to identify a region in the reference frame which corresponds substantially with the current macroblock. According to the invention, the block-matching process is performed to sub-pixel resolution in a manner that depends on the implementation of the encoder 700 and the degree of before-hand sub-pixel interpolation performed. However, the basic principle behind the block-matching process is similar in all cases. Specifically, motion estimation block 760 performs block-matching by

36

calculating difference values (e.g. sums of absolute differences) representing the difference in pixel values between the macroblock of the current frame under examination and candidate best-matching regions of pixels/sub-pixels in the reference frame. A difference value is produced for all possible offsets (e.g. quarter- or one eighth sub-pixel precision x, y displacements) between the macroblock of the current frame and candidate test region within a predefined search region of the reference frame and motion estimation block 760 determines the smallest calculated difference value. The offset between the macroblock in the current frame and the candidate test region of pixel values/sub-pixel values in the reference frame that yields the smallest difference value defines the motion vector for the macroblock in question. In certain embodiments of the invention, an initial estimate for the motion vector having unit pixel precision is first determined and then refined to a finer level of sub-pixel precision, as described in the foregoing.

In embodiments of the encoder in which before-hand sub-pixel value interpolation is not performed, all sub-pixel values required in the block matching process are calculated in on-demand sub-pixel value interpolation block 750. Motion estimation block 760 controls on-demand sub-pixel value interpolation block 750 to calculate each sub-pixel value needed in the block-matching process in an on-demand fashion, as and when it is required. In this case, motion estimation block 760 may be implemented so as to perform block-matching as a one-step process, in which case a motion vector with the desired sub-pixel resolution is sought directly, or it may be implemented so as to perform block-matching as a two step process. If the two-step process is adopted, the first step may comprise a search for e.g. a full or half-pixel resolution motion vector and the second step is performed in order to refine the motion vector to the desired sub-pixel resolution. As block matching is an exhaustive process, in which blocks of $n \times m$ pixels in the current frame are compared one-by-one with blocks of $n \times m$ pixels or sub-pixels in the interpolated reference frame, it should be appreciated that a sub-pixel calculated in an on-demand fashion by the on-demand pixel interpolation block 750 may need to be calculated multiple times as successive difference values are determined. In a video encoder, this approach is not the most efficient possible in terms of computational complexity/burden.

In embodiments of the encoder which use only before-hand sub-pixel value interpolation, block-matching may be performed as a one step process, as all sub-pixel values of the reference frame required to determine a motion vector with the desired sub-pixel resolution are calculated before-hand in block 730 and stored in frame store 740. Thus, they are directly available for use in the block-matching process and can be retrieved as required from frame store 740 by motion estimation block 760. However, even in the case where all sub-pixel values are available from frame store 740, it is still more computationally efficient to perform block-matching as a two-step process, as fewer difference calculations are required. It should be appreciated that while full before-hand sub-pixel value interpolation reduces computational complexity in the encoder, it is not the most efficient approach in terms of memory consumption.

In embodiments of the encoder in which both before-hand and on-demand sub-pixel value interpolation are used, motion estimation block 760 is implemented in such a way that it can retrieve sub-pixel values previously calculated in before-hand sub-pixel value interpolation block 730 and stored in frame store 740 and further control on-demand sub-pixel value interpolation block 750 to calculate any additional sub-pixel values that may be required. The block-

matching process may be performed as a one-step or a two-step process. If a two-step implementation is used, before-hand calculated sub-pixel values retrieved from frame store 740 may be used in the first step of the process and the second step may be implemented so as to use sub-pixel values calculated by on-demand sub-pixel value interpolation block 750. In this case, certain sub-pixel values used in the second step of the block matching process may need to be calculated multiple times as successive comparisons are made, but the number of such duplicate calculations is significantly less than if before-hand sub-pixel value calculation is not used. Furthermore, memory consumption is reduced with respect to embodiments in which only before-hand sub-pixel value interpolation is used.

Once the motion estimation block 760 has produced a motion vector for the macroblock of the current frame under examination, it outputs the motion vector to the motion field coding block 770. Motion field coding block 770 then approximates the motion vector received from motion estimation block 760 using a motion model. The motion model generally comprises a set of basis functions. More specifically, the motion field coding block 770 represents the motion vector as a set of coefficient values (known as motion coefficients) which, when multiplied by the basis functions, form an approximation of the motion vector. The motion coefficients 724 are passed from motion field coding block 770 to motion compensated prediction block 780. Motion compensated prediction block 780 also receives the pixel/sub-pixel values of the best-matching candidate test region of the reference frame identified by motion estimation block 760. In FIG. 7, these values are shown to be passed via line 729 from on-demand sub-pixel interpolation block 750. In alternative embodiments of the invention, the pixel values in question are provided from the motion estimation block 760 itself.

Using the approximate representation of the motion vector generated by motion field coding block 770 and the pixel/sub-pixel values of the best-matching candidate test region, motion compensated prediction block 780 produces a macroblock of predicted pixel values. The macroblock of predicted pixel values represents a prediction for the pixel values of the current macroblock generated from the interpolated reference frame. The macroblock of predicted pixel values is passed to the combiner 716 where it is subtracted from the new current frame in order to produce prediction error information 723 for the macroblock, as described in the foregoing.

The motion coefficients 724 formed by motion field coding block are also passed to the MUX/DMUX unit 790, where they are combined with prediction error information 723 for the macroblock in question and possible control information from control manager 720 to form an encoded video stream 725 for transmission to a receiving terminal.

Operation of a video decoder 800 according to the invention will now be described. Referring to FIG. 8, the decoder 800 comprises a demultiplexing unit (MUX/DMUX) 810, which receives the encoded video stream 725 from the encoder 700 and demultiplexes it, an inverse quantiser 820, an inverse DCT transformer 830, a motion compensated prediction block 840, a frame store 850, a combiner 860, a control manager 870, an output 880, before-hand sub-pixel value interpolation block 845 and on-demand sub-pixel interpolation block 890 associated with the motion compensated prediction block 840. In practice the control manager 870 of the decoder 800 and the control manager 720 of the encoder 700 may be the same processor. This may be the case if the encoder 700 and decoder 800 are part of the same video codec.

FIG. 8 shows an embodiment in which a combination of before-hand and on-demand sub-pixel value interpolation is used in the decoder. In other embodiments, only before-hand sub-pixel value interpolation is used, in which case decoder 800 does not include on-demand sub-pixel value interpolation block 890. In a preferred embodiment of the invention, no before-hand sub-pixel value interpolation is used in the decoder and therefore before-hand sub-pixel value interpolation block 845 is omitted from the decoder architecture. If both before-hand and on-demand sub-pixel value interpolation are performed, the decoder comprises both blocks 845 and 890.

The control manager 870 controls the operation of the decoder 800 in response to whether an INTRA or an INTER frame is being decoded. An INTRA/INTER trigger control signal, which causes the decoder to switch between decoding modes is derived, for example, from picture type information provided in the header portion of each compressed video frame received from the encoder. The INTRA/INTER trigger control signal is passed to control manager 870 via control line 815, together with other video codec control signals demultiplexed from the encoded video stream 725 by the MUX/DMUX unit 810.

When an INTRA frame is decoded, the encoded video stream 725 is demultiplexed into INTRA coded macroblocks and control information. No motion vectors are included in the encoded video stream 725 for an INTRA coded frame. The decoding process is performed macroblock-by-macroblock. When the encoded information 723 for a macroblock is extracted from video stream 725 by MUX/DMUX unit 810, it is passed to inverse quantiser 820. The control manager controls inverse quantiser 820 to apply a suitable level of inverse quantisation to the macroblock of encoded information, according to control information provided in video stream 725. The inverse quantised macroblock is then inversely transformed in the inverse DCT transformer 830 to form a decoded block of image information. Control manager 870 controls combiner 860 to prevent any reference information being used in the decoding of the INTRA coded macroblock. The decoded block of image information is passed to the video output 880 of the decoder.

In embodiments of the decoder which employ before-hand sub-pixel value interpolation, the decoded block of image information (i.e. pixel values) produced as a result of the inverse quantisation and inverse transform operations performed in blocks 820 and 830 is passed to before-hand sub-pixel value interpolation block 845. Here, sub-pixel value interpolation is performed according to the method of the invention, the degree of before-hand sub-pixel value interpolation applied being determined by the details of the decoder implementation. In embodiments of the invention in which on-demand sub-pixel value interpolation is not performed, before-hand sub-pixel value interpolation block 845 interpolates all sub-pixel values. In embodiments that use a combination of before-hand and on-demand sub-pixel value interpolation, before-hand sub-pixel value interpolation block 845 interpolates a certain sub-set of sub-pixel values. This may comprise, for example, all sub-pixels at half-pixel locations, or a combination of sub-pixels at half-pixel and one quarter-pixel locations. In any case, after before-hand sub-pixel value interpolation, the interpolated sub-pixel values are stored in frame store 850, together with the original decoded pixel values. As subsequent macroblocks are decoded, before-hand interpolated and stored, a decoded frame, at least partially interpolated to sub-pixel resolution is progressively assembled in the frame store 850 and becomes available for use as a reference frame for motion compensated prediction.

In embodiments of the decoder which do not employ before-hand sub-pixel value interpolation, the decoded block of image information (i.e. pixel values) produced as a result of the inverse quantisation and inverse transform operations performed on the macroblock in blocks **820** and **830** is passed directly to frame store **850**. As subsequent macroblocks are decoded and stored, a decoded frame, having unit pixel resolution is progressively assembled in the frame store **850** and becomes available for use as a reference frame for motion compensated prediction.

When an INTER frame is decoded, the encoded video stream **725** is demultiplexed into encoded prediction error information **723** for each macroblock of the frame, associated motion coefficients **724** and control information. Again, the decoding process is performed macroblock-by-macroblock. When the encoded prediction error information **723** for a macroblock is extracted from the video stream **725** by MUX/DMUX unit **810**, it is passed to inverse quantiser **820**. Control manager **870** controls inverse quantiser **820** to apply a suitable level of inverse quantisation to the macroblock of encoded prediction error information, according to control information received in video stream **725**. The inverse quantised macroblock of prediction error information is then inversely transformed in the inverse DCT transformer **830** to yield decoded prediction error information for the macroblock.

The motion coefficients **724** associated with the macroblock in question are extracted from the video stream **725** by MUX/DMUX unit **810** and passed to motion compensated prediction block **840**, which reconstructs a motion vector for the macroblock using the same motion model as that used to encode the INTER-coded macroblock in encoder **700**. The reconstructed motion vector approximates the motion vector originally determined by motion estimation block **760** of the encoder. The motion compensated prediction block **840** of the decoder uses the reconstructed motion vector to identify the location of a block of pixel/sub-pixel values in a prediction reference frame stored in frame store **850**. The reference frame may be, for example, a previously decoded INTRA frame, or a previously decoded INTER frame. In either case, the block of pixel/sub-pixel values indicated by the reconstructed motion vector, represents the prediction of the macroblock in question.

The reconstructed motion vector may point to any pixel or sub-pixel. If the motion vector indicates that the prediction for the current macroblock is formed from pixel values (i.e. the values of pixels at unit pixel locations), these can simply be retrieved from frame store **850**, as the values in question are obtained directly during the decoding of each frame. If the motion vector indicates that the prediction for the current macroblock is formed from sub-pixel values, these must either be retrieved from frame store **850**, or calculated in on-demand sub-pixel interpolation block **890**. Whether sub-pixel values must be calculated, or can simply be retrieved from the frame store, depends on the degree of before-hand sub-pixel value interpolation used in the decoder.

In embodiments of the decoder that do not employ before-hand sub-pixel value interpolation, the required sub-pixel values are all calculated in on-demand sub-pixel value interpolation block **890**. On the other hand, in embodiments in which all sub-pixel values are interpolated before-hand, motion compensated prediction block **840** can retrieve the required sub-pixel values directly from the frame store **850**. In embodiments that use a combination before-hand and on-demand sub-pixel value interpolation, the action required in order to obtain the required sub-pixel values depends on which sub-pixel values are interpolated before-hand. Taking

as an example an embodiment in which all sub-pixel values at half-pixel locations are calculated before-hand, it is evident that if a reconstructed motion vector for a macroblock points to a pixel at unit location or a sub-pixel at half-pixel location, all the pixel or sub-pixel values required to form the prediction for the macroblock are present in the frame store **850** and can be retrieved from there by motion compensated prediction block **840**. If, however, the motion vector indicates a sub-pixel at a quarter-pixel location, the sub-pixels required to form the prediction for the macroblock are not present in frame store **850** and are therefore calculated in on-demand sub-pixel value interpolation block **890**. In this case, on-demand sub-pixel value interpolation block **890** retrieves any pixel or sub-pixel required to perform the interpolation from frame store **850** and applies the interpolation method described below. Sub-pixel values calculated in on-demand sub-pixel value interpolation block **890** are passed to motion compensated prediction block **840**.

Once a prediction for a macroblock has been obtained, the prediction (that is, a macroblock of predicted pixel values) is passed from motion compensated prediction block **840** to combiner **860** where it combined with the decoded prediction error information for the macroblock to form a reconstructed image block which, in turn, is passed to the video output **880** of the decoder.

It should be appreciated that in practical implementations of encoder **700** and decoder **800**, the extent to which frames are before-hand sub-pixel interpolated, and thus the amount of on-demand sub-pixel value interpolation that is performed, can be chosen according to, or dictated by, the hardware implementation of the video encoder **700**, or the environment in which it is intended to be used. For example, if the memory available to the video encoder is limited, or memory must be reserved for other functions, it is appropriate to limit the amount of before-hand sub-pixel value interpolation that is performed. In other cases, where the microprocessor performing the video encoding operation has limited processing capacity, e.g. the number of operations per second that can be executed is comparatively low, it is more appropriate to restrict the amount of on-demand sub-pixel value interpolation that is performed. In a mobile communications environment, for example, when video encoding and decoding functionality is incorporated in a mobile telephone or similar wireless terminal for communication with a mobile telephone network, both memory and processing power may be limited. In this case a combination of before-hand and on-demand sub-pixel value interpolation may be the best choice to obtain an efficient implementation in the video encoder. In video decoder **800**, use of before-hand sub-pixel value is generally not preferred, as it typically results in the calculation of many sub-pixel values that are not actually used in the decoding process. However, it should be appreciated that although different amounts of before-hand and on-demand interpolation can be used in the encoder and decoder in order to optimise the operation of each, both encoder and decoder can be implemented so as to use the same division between before-hand and on-demand sub-pixel value interpolation.

Although the foregoing description does not describe the construction of bi-directionally predicted frames (B-frames) in the encoder **700** and the decoder **800**, it should be understood that in embodiments of the invention, such a capability may be provided. Provision of such capability is considered within the ability of one skilled in the art.

An encoder **700** or a decoder **800** according to the invention can be realised using hardware or software, or using a suitable combination of both. An encoder or decoder implemented in software may be, for example, a separate program or a soft-

41

ware building block that can be used by various programs. In the above description and in the drawings, the functional blocks are represented as separate units, but the functionality of these blocks can be implemented, for example, in one software program unit.

The encoder **700** and decoder **800** can further be combined in order to form a video codec having both encoding and decoding functionality. In addition to being implemented in a multimedia terminal, such a codec may also be implemented in a network. A codec according to the invention may be a computer program or a computer program element, or it may be implemented at least partly using hardware.

The sub-pixel interpolation method used in the encoder **700** and decoder **800** according to the invention now be described in detail. The method will first be introduced at a general conceptual level and then two preferred embodiments will be described. In the first preferred embodiment, sub-pixel value interpolation is performed to $\frac{1}{4}$ pixel resolution and in the second the method is extended to $\frac{1}{8}$ pixel resolution.

It should be noted that interpolation must produce identical values in the encoder and the decoder, but its implementation should be optimized for both entities separately. For example, in an encoder according to the first embodiment of the invention in which sub-pixel value interpolation is performed to $\frac{1}{4}$ pixel resolution, it is most efficient to calculate $\frac{1}{2}$ resolution pixels before-hand and to calculate values for $\frac{1}{4}$ resolution sub-pixels in an on-demand fashion, only when they are needed during motion estimation. This has the effect of limiting memory usage while keeping the computational complexity/burden at an acceptable level. In the decoder, on the other hand, it is advantageous not to pre-calculate any of the sub-pixels. Therefore, it should be appreciated that a preferred embodiment of the decoder does not include before-hand sub-pixel value interpolation block **845** and all sub-pixel value interpolation is performed in on-demand sub-pixel value interpolation block **890**.

In the description of the interpolation method provided below, references are made to the pixel positions depicted in FIG. **14a**. In this figure pixels labelled A represent original pixels (that is, pixels residing at unit horizontal and vertical locations). Pixels labelled with other letters represent sub-pixels that are to be interpolated. The description that follows will adhere to the previously introduced conventions regarding the description of pixel and sub-pixel locations.

Next, the steps required to interpolate all sub-pixel positions are described:

Values for the $\frac{1}{2}$ resolution sub-pixels labelled b are obtained by first calculating an intermediate value b using a Kth order filter, according to:

$$b = \sum_{i=1}^K x_i A_i \quad (9)$$

where x_i is a vector of filter coefficients, A_i is a corresponding vector of original pixel values A situated at unit horizontal and unit vertical locations, and K is an integer which defines the order of the filter. Thus, equation 9 can be re-expressed as:

$$b = x_1 A_1 + x_2 A_2 + x_3 A_3 + \dots + x_{K-1} A_{K-1} + x_K A_K \quad (10)$$

The values of the filter coefficients x_i and the order of the filter K may vary from embodiment to embodiment. Equally, different coefficient values may be used in the calculation of different sub-pixels within an embodiment. In other embodiments, the values of filter coefficients x_i and the order of the

42

filter may depend on which of the $\frac{1}{2}$ resolution b sub-pixels is being interpolated. Pixels A_i are disposed symmetrically with respect to the $\frac{1}{2}$ resolution sub-pixel b being interpolated and are the closest neighbours of that sub-pixel. In the case of the $\frac{1}{2}$ resolution sub-pixel b situated at half horizontal and unit vertical location, pixels A_i are disposed horizontally with respect to b (as shown in FIG. **14b**). If the $\frac{1}{2}$ resolution sub-pixel b situated at unit horizontal and half vertical location is being interpolated, pixels A_i are disposed vertically with respect to b (as shown in FIG. **14c**).

A final value for $\frac{1}{2}$ resolution sub-pixel b is calculated by dividing intermediate value b by a constant scale₁, truncating it to obtain an integer number and clipping the result to lie in the range $[0, 2^n - 1]$. In alternative embodiments of the invention rounding may be performed instead of truncation. Preferably, constant scale₁ is chosen to be equal to the sum of filter coefficients x_i .

A value for the $\frac{1}{2}$ resolution sub-pixel labelled c is also obtained by first calculating an intermediate value c using an Mth order filter, according to:

$$c = \sum_{i=1}^M y_i b_i \quad (11)$$

where y_i is a vector of filter coefficients, b_i is a corresponding vector of intermediate values b_i in the horizontal or vertical direction. i.e.:

$$c = y_1 b_1 + y_2 b_2 + y_3 b_3 + \dots + y_{M-1} b_{M-1} + y_M b_M \quad (12)$$

The values of the filter coefficients y_i and the order of the filter M may vary from embodiment to embodiment. Equally, different coefficient values may be used in the calculation of different sub-pixels within an embodiment. Preferably, the b values are intermediate values for $\frac{1}{2}$ resolution sub-pixels b which are disposed symmetrically with respect to $\frac{1}{2}$ resolution sub-pixel c and are the closest neighbours of sub-pixel c. In an embodiment of the invention, the $\frac{1}{2}$ resolution sub-pixels b are disposed horizontally with respect to sub-pixel c, in an alternative embodiment they are disposed vertically with respect to sub-pixel c.

A final value of $\frac{1}{2}$ resolution sub-pixel c is computed by dividing intermediate value c by a constant scale₂, truncating it to obtain an integer number and clipping the result to lie in the range $[0, 2^n - 1]$. In alternative embodiments of the invention rounding may be performed instead of truncation. Preferably, constant scale₂ is equal to scale₁ * scale₁.

It should be noted that the use of intermediate values b in the horizontal direction leads to the same result as using intermediate values b in the vertical direction.

There are two alternatives for interpolating values for the $\frac{1}{4}$ resolution sub-pixels labelled h. Both involve linear interpolation along a diagonal line linking $\frac{1}{2}$ resolution sub-pixels neighbouring the $\frac{1}{4}$ resolution sub-pixel h being interpolated. In a first embodiment, a value for sub-pixel h is calculated by averaging the values of the two $\frac{1}{2}$ resolution sub-pixels b closest to sub-pixel h. In a second embodiment, a value for sub-pixel h is calculated by averaging the values of the closest pixel A and the closest $\frac{1}{2}$ resolution sub-pixel c. It should be appreciated that this provides the possibility of using different combinations of diagonal interpolations to determine the values for sub-pixels h within the confines of different groups of 4 image pixels A. However, it should also be realised that the same combination should be used in both the encoder and the decoder in order to produce identical interpolation results.

FIG. 15 depicts 4 possible choices of diagonal interpolation for sub-pixels h in adjacent groups of 4 pixels within an image. Simulations in the TML environment have verified that both embodiments result in similar compression efficiency. The second embodiment has higher complexity, since calculation of sub-pixel c requires calculation of several intermediate values. Therefore the first embodiment is preferred.

Values for $\frac{1}{4}$ resolution sub-pixels labelled d and g are calculated from the values of their nearest horizontal neighbours using linear interpolation. In other words, a value for $\frac{1}{4}$ resolution sub-pixel d is obtained by averaging values of its nearest horizontal neighbours, original image pixel A and $\frac{1}{2}$ resolution sub-pixel b. Similarly, a value for $\frac{1}{4}$ resolution sub-pixel g is obtained by taking the average of its two nearest horizontal neighbours, $\frac{1}{2}$ resolution sub-pixels b and c.

Values for $\frac{1}{4}$ resolution sub-pixels labelled e, f and i are calculated from the values of their nearest neighbours in the vertical direction using linear interpolation. More specifically, a value for $\frac{1}{4}$ resolution sub-pixel e is obtained by averaging the values of its two nearest vertical neighbours, original image pixel A and $\frac{1}{2}$ resolution sub-pixel b. Similarly, a value for $\frac{1}{4}$ resolution sub-pixel f is obtained by taking the average of its two nearest vertical neighbours, $\frac{1}{2}$ resolution sub-pixels b and c. In an embodiment of the invention, a value for $\frac{1}{4}$ resolution sub-pixel i is obtained in manner identical to that just described in connection with $\frac{1}{4}$ resolution sub-pixel f. However, in an alternative embodiment of the invention, and in common with the H.26 test models TML5 and TML6 previously described, $\frac{1}{4}$ resolution sub-pixel i is determined using the values of the four closest original image pixels, according to $(A_1 + A_2 + A_3 + A_4 + 2)/4$.

It should also be noted that in all cases where an average involving pixel and/or sub-pixel values is determined, the average may be formed in any appropriate manner. For example, the value for $\frac{1}{4}$ resolution sub-pixel d can be defined as $d = (A + b)/2$ or as $d = (A + b + 1)/2$. The addition of 1 to the sum of values for pixel A and $\frac{1}{2}$ resolution sub-pixel b has the effect of causing any rounding or truncation operation subsequently applied to round or truncate the value for d to the next highest integer value. This is true for any sum of integer values and may be applied to any of the averaging operations performed according to the method of the invention in order to control rounding or truncation effects.

It should be noted that the sub-pixel value interpolation method according to the invention provides advantages over each of TML5 and TML6.

In contrast to TML5, in which the values of some of the $\frac{1}{4}$ resolution sub-pixels depend on previously interpolated values obtained for other $\frac{1}{4}$ resolution sub-pixels, in the method according to the invention, all $\frac{1}{4}$ resolution sub-pixels are calculated from original image pixels or $\frac{1}{2}$ resolution sub-pixel positions using linear interpolation. Thus, the reduction in precision of those $\frac{1}{4}$ resolution sub-pixel values which occurs in TML5 due to the intermediate truncation and clipping of the other $\frac{1}{4}$ resolution sub-pixels from which they are calculated, does not take place in the method according to the invention. In particular, referring to FIG. 14a, $\frac{1}{4}$ resolution sub-pixels h (and sub-pixel i in one embodiment of the invention) are interpolated diagonally in order to reduce dependency on other $\frac{1}{4}$ -pixels. Furthermore, in the method according to the invention, the number of calculations (and therefore the number of processor cycles) required to obtain a value for those $\frac{1}{4}$ resolution sub-pixels in the decoder is reduced compared with TML5. Additionally, the calculation of any $\frac{1}{4}$ resolution sub-pixel value requires a number of calculations which is substantially similar to the number of calculations required to determine any other $\frac{1}{4}$ resolution sub-pixel value.

More specifically, in a situation where the required $\frac{1}{2}$ resolution sub-pixel values are already available, e.g. they have been calculated before-hand, the number of calculations required to interpolate a $\frac{1}{4}$ resolution sub-pixel value from the pre-calculated $\frac{1}{2}$ resolution sub-pixel values is the same as the number of calculations required to calculate any other $\frac{1}{4}$ resolution sub-pixel value from the available $\frac{1}{2}$ resolution sub-pixel values.

In comparison with TML6, the method according to the invention does not require high precision arithmetic to be used in the calculation of all sub-pixels. Specifically, as all of the $\frac{1}{4}$ resolution sub-pixel values are calculated from original image pixels or $\frac{1}{2}$ resolution sub-pixel values using linear interpolation, lower precision arithmetic can be used in their interpolation. Consequently, in hardware implementations of the inventive method, for example in an ASIC (Application Specific Integrated Circuit), the use of lower precision arithmetic reduces the number of components (e.g. gates) that must be devoted to the calculation of $\frac{1}{4}$ resolution sub-pixel values. This, in turn, reduces the overall area of silicon that must be dedicated to the interpolation function. As the majority of sub-pixels are, in fact, $\frac{1}{4}$ resolution sub-pixels (12 out of the 15 sub-pixels illustrated in FIG. 14a), the advantage provided by the invention in this respect is particularly significant. In software implementations, where sub-pixel interpolation is performed using the standard instruction set of a general purpose CPU (Central Processor Unit) or using a DSP (Digital Signal Processor), a reduction in the precision of the arithmetic required generally leads to an increase in the speed at which the calculations can be performed. This is particularly advantageous in 'low cost' implementations, in which it is desirable to use a general purpose CPU rather than any form of ASIC.

The method according to the invention provides still further advantages compared with TML5. As mentioned previously, in the decoder only 1 out of the 15 sub-pixel positions is required at any given time, namely that which is indicated by received motion vector information. Therefore, it is advantageous if the value of a sub-pixel in any sub-pixel location can be calculated with the minimum number of steps that result in a correctly interpolated value. The method according to the invention provides this capability. As mentioned in the detailed description provided above, $\frac{1}{2}$ resolution sub-pixel c can be interpolated by filtering in either the vertical or the horizontal direction, the same value being obtained for c regardless of whether horizontal or vertical filtering is used. The decoder can therefore take advantage of this property when calculating values for $\frac{1}{4}$ resolution sub-pixels f and g in such a way as to minimise the number of operations required in order to obtain the required values. For example, if the decoder requires a value for $\frac{1}{4}$ resolution sub-pixel f, $\frac{1}{2}$ resolution sub-pixel c should be interpolated in the vertical direction. If a value is required for $\frac{1}{4}$ resolution sub-pixel g, it is advantageous to interpolate a value for c in the horizontal direction. Thus, in general, it can be said that the method according to the invention provides flexibility in the way in which values are derived for certain $\frac{1}{4}$ resolution sub-pixels. No such flexibility is provided in TML5.

Two specific embodiments will now be described in detail. The first represents a preferred embodiment for calculating sub-pixels with up to $\frac{1}{4}$ pixel resolution, while in the second, the method according to the invention is extended to the calculation of values for sub-pixels having up to $\frac{1}{8}$ pixel resolution. For both embodiments a comparison is provided between the computational complexity/burden resulting from use of the method according to the invention and that which

would result from use of the interpolation methods according to TML5 and TML6 in equivalent circumstances.

The preferred embodiment for interpolating sub-pixels at $\frac{1}{4}$ pixel resolution will be described with reference to FIGS. 14a, 14b and 14c. In the following, it will be assumed that all image pixels and final interpolated values for sub-pixels are represented with 8-bits.

Calculation of $\frac{1}{2}$ resolution sub-pixels at i) half unit horizontal and unit vertical locations and ii) unit horizontal and half unit vertical locations.

1. A value for the sub-pixel at half unit horizontal and unit vertical location, that is $\frac{1}{2}$ resolution sub-pixel b in FIG. 14a, is obtained by first calculating intermediate value $b = (A_1 - 5A_2 + 20A_3 + 20A_4 - 5A_5 + A_6)$ using the values of the six pixels (A_1 to A_6) which are situated at unit horizontal and unit vertical locations in either the row or the column of pixels containing b and which are disposed symmetrically about b, as shown in FIGS. 14b and 14c. A final value for $\frac{1}{2}$ resolution sub-pixel b is calculated as $(b+16)/32$ where the operator / denotes division with truncation. The result is clipped to lie in the range [0, 255].

Calculation of $\frac{1}{2}$ resolution sub-pixels at half unit horizontal and half unit vertical locations.

2. A value for the sub-pixel at half unit horizontal and half unit vertical location, that is $\frac{1}{2}$ resolution sub-pixel c in FIG. 14a, is calculated as $c = (b_1 - 5b_2 + 20b_3 + 20b_4 - 5b_5 + b_6 + 512)/1024$ using the intermediate values b for the six closest $\frac{1}{2}$ resolution sub-pixels which are situated in either the row or the column of sub-pixels containing c and which are disposed symmetrically about c. Again, operator / denotes division with truncation and the result is clipped to lie in the range [0, 255]. As previously explained, using intermediate values b for $\frac{1}{2}$ resolution sub-pixels b in the horizontal direction leads to the same result as using intermediate values b for $\frac{1}{2}$ resolution sub-pixels b in the vertical direction. Thus, in an encoder according to the invention, the direction for interpolating $\frac{1}{2}$ resolution sub-pixels b can be chosen according to a preferred mode of implementation. In a decoder according to the invention, the direction for interpolating sub-pixels b is chosen according to which, if any, $\frac{1}{4}$ resolution sub-pixels will be interpolated using the result obtained for $\frac{1}{2}$ resolution sub-pixel c.

Calculation of $\frac{1}{4}$ resolution sub-pixels at i) quarter unit horizontal and unit vertical locations; ii) quarter unit horizontal and half unit vertical locations; iii) unit horizontal and quarter unit vertical locations; and iv) half unit horizontal and quarter unit vertical locations.

3. Values for $\frac{1}{4}$ resolution sub-pixels d, situated at quarter unit horizontal and unit vertical locations are calculated according to $d = (A+b)/2$ using the nearest original image pixel A and the closest $\frac{1}{2}$ resolution sub-pixel b in the horizontal direction. Similarly, values for $\frac{1}{2}$ resolution sub-pixels g, situated at quarter unit horizontal and half unit vertical locations are calculated according to $g = (b+c)/2$ using the two nearest $\frac{1}{2}$ resolution sub-pixels in the horizontal direction. In a similar manner, values for $\frac{1}{4}$ resolution sub-pixels e, situated at unit horizontal and quarter unit vertical locations, are calculated according to $e = (A+b)/2$ using the nearest original image pixel A and the closest $\frac{1}{2}$ resolution sub-pixel b in the vertical direction. Values for $\frac{1}{4}$ resolution sub-pixels f, situated at half unit horizontal and quarter unit vertical locations, are determined from $f = (b+c)/2$ using the two nearest $\frac{1}{2}$ resolution sub-pixel in the vertical direction. In all cases, operator / denotes division with truncation.

Calculation of $\frac{1}{4}$ resolution sub-pixels at quarter unit horizontal and quarter unit vertical locations.

4. Values for $\frac{1}{4}$ resolution sub-pixels h, situated at quarter unit horizontal and quarter unit vertical locations are calculated according to $h = (b_1 + b_2)/2$, using the two nearest $\frac{1}{2}$ resolution sub-pixels b in the diagonal direction. Again, operator / denotes division with truncation.

5. A value for the $\frac{1}{4}$ resolution sub-pixel labeled i is computed from $i = (A_1 + A_2 + A_3 + A_4 + 2)/4$ using the four nearest original pixels A. Once more, operator / denotes division with truncation.

10 An analysis of the computational complexity of the first preferred embodiment of the invention will now be presented.

In the encoder, it is likely that the same sub-pixel values will be calculated multiple times. Therefore, and as previously explained, the complexity of the encoder can be reduced by pre-calculating all sub-pixel values and storing them in memory. However, this solution increases memory usage by a large margin. In the preferred embodiment of the invention, in which motion vector accuracy is $\frac{1}{4}$ pixel resolution in both the horizontal and vertical dimensions, storing pre-calculated sub-pixel values for the whole image requires 16 times the memory required to store the original, non-interpolated image. To reduce memory usage, all $\frac{1}{2}$ resolution sub-pixels can be interpolated before-hand and $\frac{1}{4}$ resolution sub-pixels can be calculated on-demand, that is, only when they are needed. According to the method of the invention, on-demand interpolation of values for $\frac{1}{4}$ resolution sub-pixels only requires linear interpolation from $\frac{1}{2}$ resolution sub-pixels. Four times the original picture memory is required to store the pre-calculated $\frac{1}{2}$ resolution sub-pixels since only 8 bits are necessary to represent them.

However, if the same strategy of pre-calculating all $\frac{1}{2}$ resolution sub-pixels using before-hand interpolation is used in conjunction with the direct interpolation scheme of TML6, the memory requirements increase to 9 times the memory required to store the original non-interpolated image. This results from the fact that a larger number of bits is required to store the high precision intermediate values associated with each $\frac{1}{2}$ resolution sub-pixel in TML6. In addition, the complexity of sub-pixel interpolation during motion estimation is higher in TML6, since scaling and clipping has to be performed for every $\frac{1}{2}$ and $\frac{1}{4}$ sub-pixel location.

In the following, the complexity of the sub-pixel value interpolation method according to the invention, when applied in a video decoder, is compared with that of the interpolation schemes used in TML5 and TML6. Throughout the analysis which follows, it is assumed that in each method the interpolation of any sub-pixel value is performed using only the minimum number of steps required to obtain a correctly interpolated value. It is further assumed that each method is implemented in a block based manner, that is, intermediate values common for all the sub-pixels to be interpolated in a particular $N \times M$ block are calculated only once. An illustrative example is provided in FIG. 16. Referring to FIG. 16, it can be seen that in order to calculate a 4×4 block of $\frac{1}{2}$ resolution sub-pixels c, a 9×4 block of $\frac{1}{2}$ resolution sub-pixels b is first calculated.

Compared with the sub-pixel value interpolation method used in TML5, the method according to the invention has a lower computational complexity for the following reasons:

1. Unlike the sub-pixel value interpolation scheme used in TML5, according to the method of the invention, a value for $\frac{1}{2}$ resolution sub-pixel c can be obtained by filtering in either the vertical or the horizontal direction. Thus, in order to reduce the number of operations, $\frac{1}{2}$ resolution sub-pixel c can be interpolated in the vertical direction if a value for $\frac{1}{4}$ resolution sub-pixel f is required and in the horizontal direction if a value for $\frac{1}{4}$ resolution sub-pixel g is

required. As an example, FIG. 17 shows all the $\frac{1}{2}$ resolution sub-pixel values that must be calculated in order to interpolate values for $\frac{1}{4}$ resolution sub-pixels g in an image block defined by 4×4 original image pixels using the interpolation method of TML5 (FIG. 17a) and using the method according to the invention (FIG. 17b). In this example, the sub-pixel value interpolation method according to TML5 requires a total of $88\frac{1}{2}$ resolution sub-pixels to be interpolated, while the method according to the invention requires the calculation of $72\frac{1}{2}$ resolution sub-pixels. As can be seen from FIG. 17b, according to the invention, $\frac{1}{2}$ resolution sub-pixels c are interpolated in the horizontal direction in order to reduce the number of calculations required.

2. According to the method of the invention, $\frac{1}{4}$ resolution sub-pixel h is calculated by linear interpolation from its two closest neighbouring $\frac{1}{2}$ resolution sub-pixels in the diagonal direction. The respective numbers of $\frac{1}{2}$ resolution sub-pixels that must be calculated in order to obtain values for $\frac{1}{4}$ resolution sub-pixels h within a 4×4 block of original image pixels using the sub-pixel value interpolation method according to TML5 and the method according to the invention are shown in FIGS. 18(a) and 18(b), respectively. Using the method according to TML5 it is necessary to interpolate a total of $56\frac{1}{2}$ resolution sub-pixels, while according to the method of the invention it is necessary to interpolate $40\frac{1}{2}$ resolution sub-pixels.

Table 1 summarizes the decoder complexities of the three sub-pixel value interpolation methods considered here, that according to TML5, the direct interpolation method used in TML6 and the method according to the invention. Complexity is measured in terms of the number of 6-tap filtering and linear interpolation operations performed. It is assumed that Interpolation of $\frac{1}{4}$ resolution sub-pixel i is calculated according to $i = (A_1 + A_2 + A_3 + A_4 + 2)/4$ which is bilinear interpolation and effectively comprises two linear interpolation operations. The operations needed to interpolate sub-pixel values with one 4×4 block of original image pixels are listed for each of the 15 sub-pixel positions which, for convenience of reference, are numbered according to the scheme shown in FIG. 19. Referring to FIG. 19, location 1 is the location of an original image pixel A and locations 2 to 16 are sub-pixel locations. Location 16 is the location of $\frac{1}{4}$ resolution sub-pixel i. In order to compute the average number of operations it has been assumed that the probability of a motion vector pointing to each sub-pixel position is the same. The average complexity is therefore the average of the 15 sums calculated for each sub-pixel location and the single full-pixel location.

TABLE 1

Complexity of $\frac{1}{4}$ resolution sub-pixel interpolation in TML5, TML6 and the method according to the invention.						
Location	TML5		TML6		Inventive Method	
	linear.	6-tap	linear.	6-tap	linear.	6-tap
1	0	0	0	0	0	0
3, 9	0	16	0	16	0	16
2, 4, 5, 13	16	16	0	16	16	16
11	0	52	0	52	0	52
7, 15	16	52	0	52	16	52
10, 12	16	68	0	52	16	52
6, 8, 14	48	68	0	52	16	32
16	32	0	32	0	32	0
Average	19	37	2	32	13	28.25

It can be seen from Table 1 that the method according to the invention requires fewer 6-tap filter operations than the sub-

pixel value interpolation method according to TML6 and only a few additional linear interpolation operations. Since 6-tap filter operations are much more complex than linear interpolation operations the complexity of the two methods is similar. The sub-pixel value interpolation method according to TML5 has a considerably higher complexity.

The preferred embodiment for interpolating sub-pixels with up to $\frac{1}{8}$ pixel resolution will now be described with reference to FIGS. 20, 21 and 22.

FIG. 20 presents the nomenclature used to describe pixels, $\frac{1}{2}$ resolution sub-pixels, $\frac{1}{4}$ resolution sub-pixels and $\frac{1}{8}$ resolution sub-pixels in this extended application of the method according to the invention.

1. Values for the $\frac{1}{2}$ resolution and $\frac{1}{4}$ resolution sub-pixels labelled b^1 , b^2 and b^3 in FIG. 20 are obtained by first calculating intermediate values $b^1 = (-3A_1 + 12A_2 - 37A_3 + 229A_4 + 71A_5 - 21A_6 + 6A_7 - A_8)$; $b^2 = (-3A_1 + 12A_2 - 39A_3 + 158A_4 + 158A_5 - 39A_6 + 12A_7 - 3A_8)$; and $b^3 = (-A_1 + 6A_2 - 21A_3 + 71A_4 + 229A_5 - 37A_6 + 13A_7 - 3A_8)$ using the values of the eight nearest image pixels (A_1 to A_8) situated at unit horizontal and unit vertical locations in either the row or the column containing b^1 , b^2 and b^3 and disposed symmetrically about $\frac{1}{2}$ resolution sub-pixel b^2 . The asymmetries in the filter coefficients used to obtain intermediate values b^1 and b^3 account for the fact that pixels A_1 to A_8 are not symmetrically located with respect to $\frac{1}{4}$ resolution sub-pixels b^1 and b^3 . Final values for sub-pixels b^i , $i=1, 2, 3$ are calculated according to $b^i = (b^i + 128)/256$ where the operator / denotes division with truncation. The result is clipped to lie in the range $[0, 255]$.

2. Values for the $\frac{1}{2}$ resolution and $\frac{1}{4}$ resolution sub-pixels labelled c^{ij} , $i, j=1, 2, 3$, are calculated according to $c^{ij} = (-3b_1^j + 12b_2^j - 37b_3^j + 229b_4^j + 71b_5^j - 21b_6^j + 6b_7^j - b_8^j + 32768)/65536$; $c^{2j} = (-3b_1^j + 12b_2^j - 39b_3^j + 158b_4^j + 158b_5^j - 39b_6^j + 12b_7^j - 3b_8^j + 32768)/65536$ and $c^{3j} = (-b_1^j + 6b_2^j - 21b_3^j + 71b_4^j + 229b_5^j - 37b_6^j + 13b_7^j - 3b_8^j + 32768)/65536$ using the intermediate values b^1 , b^2 and b^3 calculated for the eight closest sub-pixels (b_1^j to b_8^j) in the vertical direction, sub-pixels b^j being situated in the column comprising the $\frac{1}{2}$ resolution and $\frac{1}{4}$ resolution sub-pixels c^{ij} being interpolated and disposed symmetrically about the $\frac{1}{2}$ resolution sub-pixel c^{2j} . The asymmetries in the filter coefficients used to obtain values for sub-pixels c^{1j} and c^{3j} account for the fact that sub-pixels b_1^j to b_8^j are not symmetrically located with respect to $\frac{1}{4}$ resolution sub-pixels c^{1j} and c^{3j} . Once more, operator / denotes division with truncation. Before the interpolated values for sub-pixels c^{ij} are stored in the frame memory they are clipped to lie in the range $[0, 255]$. In an alternative embodiment of the invention, $\frac{1}{2}$ resolution and $\frac{1}{4}$ resolution sub-pixels c^{ij} are calculated using in an analogous manner using intermediate values b^1 , b^2 and b^3 in the horizontal direction.

3. Values for $\frac{1}{8}$ resolution sub-pixels labelled d are calculated using linear interpolation from the values of their closest neighbouring image pixel, $\frac{1}{2}$ resolution or $\frac{1}{4}$ resolution sub-pixels in the horizontal or vertical direction. For example, upper leftmost $\frac{1}{8}$ resolution sub-pixel d is calculated according to $d = (A + b^1 + 1)/2$. As before, operator / indicates division with truncation.

4. Values for $\frac{1}{8}$ resolution sub-pixels labelled e and f are calculated using linear interpolation from the values of image pixels, $\frac{1}{2}$ resolution or $\frac{1}{4}$ resolution sub-pixels in the diagonal direction. For example, referring to FIG. 20, upper leftmost pixel $\frac{1}{8}$ resolution sub-pixel e is calculated according to $e = (b^1 + b^1 + 1)/2$. The diagonal direction to be used in the interpolation of each $\frac{1}{8}$ resolution sub-pixel in a first preferred embodiment of the invention, hereinafter

referred to as 'preferred method 1', is indicated in FIG. 21(a). Values for $\frac{1}{8}$ resolution sub-pixels labelled g are calculated according to $g=(A+3c^{22}+3)/4$. As always, operator / denotes division with truncation. In an alternative embodiment of the invention, hereinafter referred to as 'preferred method 2', computational complexity is further reduced by interpolating $\frac{1}{8}$ resolution sub-pixels f using linear interpolation from $\frac{1}{2}$ resolution sub-pixels b², that is, according to the relationship $f=(3b^2+b^2+2)/4$. The b² sub-pixel which is closer to f is multiplied by 3. The diagonal interpolation scheme used in this alternative embodiment of the invention is depicted in FIG. 21(b). In further alternative embodiments, different diagonal interpolation schemes can be envisaged.

It should be noted that in all cases where an average involving pixel and/or sub-pixel values is used in the determination of $\frac{1}{8}$ resolution sub-pixels, the average may be formed in any appropriate manner. The addition of 1 to the sum of values used in calculating such an average has the effect of causing any rounding or truncation operation subsequently applied to round or truncate the average in question to the next highest integer value. In alternative embodiments of the invention, the addition of 1 is not used.

As in the case of sub-pixel value interpolation to $\frac{1}{4}$ pixel resolution previously described, memory requirements in the encoder can be reduced by pre-calculating only a part of the sub-pixel values to be interpolated. In the case of sub-pixel

applied in a video decoder to calculate values for sub-pixels at up to $\frac{1}{8}$ pixel resolution, is compared with that of the interpolation schemes used in TML5 and TML6. As in the equivalent analysis for $\frac{1}{4}$ pixel resolution sub-pixel value interpolation described above, it is assumed that in each method the interpolation of any sub-pixel value is performed using only the minimum number of steps required to obtain a correctly interpolated value. It is also assumed that each method is implemented in a block based manner, such that intermediate values common for all the sub-pixels to be interpolated in a particular N×M block are calculated only once.

Table 2 summarizes complexities of the three interpolation methods. Complexity is measured in terms of the number of 8-tap filter and linear interpolation operations performed in each method. The table presents the number of operations required to interpolate each of the $63\frac{1}{8}$ resolution sub-pixels within one 4×4 block of original image pixels, each sub-pixel location being identified with a corresponding number, as illustrated in FIG. 22. In FIG. 22, location 1 is the location of an original image pixel and locations 2 to 64 are sub-pixel locations. When computing the average number of operations, it has been assumed that the probability of a motion vector pointing to each sub-pixel position is the same. The average complexity is thus the average of the 63 sums calculated for each sub-pixel location and the single full-pixel location.

TABLE 2

Complexity of $\frac{1}{8}$ resolution sub-pixel interpolation in TML5, TML6 and the method according to the invention. (Results shown separately for Preferred Method 1 and Preferred Method 2).								
Location	TML5		TML6		Preferred Method 1		Preferred Method 2	
	linear.	8-tap	linear.	8-tap	linear.	8-tap	linear.	8-tap
1	0	0	0	0	0	0	0	0
3, 5, 7, 17, 33, 49	0	16	0	16	0	16	0	16
19, 21, 23, 35, 37, 39, 51, 53, 55	0	60	0	60	0	60	0	60
2, 8, 9, 57	16	16	0	16	16	16	16	16
4, 6, 25, 41	16	32	0	16	16	32	16	32
10, 16, 58, 64	32	76	0	60	16	32	16	32
11, 13, 15, 59, 61, 63	16	60	0	60	16	60	16	60
18, 24, 34, 40, 50, 56	16	76	0	60	16	60	16	60
12, 14, 60, 62	32	120	0	60	16	32	16	32
26, 32, 42, 48	32	108	0	60	16	32	16	32
20, 22, 36, 38, 52, 54	16	120	0	60	16	76	16	76
27, 29, 31, 43, 45, 47	16	76	0	60	16	76	16	76
28, 30, 44, 46	32	152	0	60	16	60	16	60
Average	64	290.25	0	197.75	48	214.75	48	192.75

value interpolation to $\frac{1}{8}$ pixel resolution, it is advantageous to calculate all $\frac{1}{2}$ resolution and $\frac{1}{4}$ resolution sub-pixels beforehand and to compute values for $\frac{1}{8}$ resolution sub-pixels in an on-demand fashion, only when they are required. When this approach is taken, both the interpolation method according to TML5 and that according to the invention require 16 times the original picture memory to store the $\frac{1}{2}$ resolution and $\frac{1}{4}$ resolution sub-pixel values. However, if the direct interpolation method according to TML6 is used in the same way, intermediate values for the $\frac{1}{2}$ resolution and $\frac{1}{4}$ pixel resolution sub-pixels must be stored. These intermediate values are represented with 32-bit precision and this results in a memory requirement 64 times that for the original, non-interpolated image.

In the following, the complexity of the sub-pixel value interpolation method according to the invention, when

As can be seen from Table 2, the number of 8-tap filtering operations performed according to preferred methods 1 and 2 are, respectively, 26% and 34% lower than the number of 8-tap filtering operation performed in the sub-pixel value interpolation method of TML5. The number of linear operations is 25% lower, in both preferred method 1 and preferred method 2, compared with TML5, but this improvement is of lesser importance compared to the reduction in 8-tap filtering operations. It can further be seen that the direct interpolation method used in TML6 has a complexity comparable that of both preferred methods 1 and 2 when used to interpolate values for $\frac{1}{8}$ resolution sub-pixels.

In view of the foregoing description it will be evident to a person skilled in the art that various modifications may be made within the scope of the invention. While a number of preferred embodiments of the invention have been described

in detail, it should be apparent that many modifications and variations thereto are possible, all of which fall within the true spirit and scope of the invention.

The invention claimed is:

1. A method for sub-pixel value interpolation to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being configured for display in rows and columns, pixel and sub-pixel locations in the rows and columns being representable mathematically within the rectangular bounded region using the co-ordinate notation $K/2^N$, $L/2^N$, K and L being positive integers having respective values between zero and 2^N , N being a positive integer greater than one and representing a particular degree of sub-pixel value interpolation, the method comprising causing an apparatus to:

interpolate a sub-pixel value for a sub-pixel having co-ordinates with odd values of K and L, according to a predetermined choice of either a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates $\frac{1}{2}$, $\frac{1}{2}$, or a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with even values of K and L, including zero, situated within a quadrant of the rectangular bounded region, the quadrant being defined by the sub-pixel having co-ordinates $\frac{1}{2}$, $\frac{1}{2}$ and the nearest neighbouring pixel;

interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an even value and L equal to zero and sub-pixels having co-ordinates with K equal to zero and L equal to an even value, used in the interpolation of the sub-pixels having co-ordinates with odd values of K and L, using weighted sums of the values of pixels located in rows and columns respectively; and

interpolate sub-pixel values for sub-pixels having co-ordinates with even values of K and L, used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of K and L, using a predetermined choice of either a weighted sum of the values of sub-pixels having co-ordinates with K equal to an even value and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, or a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions.

2. A method according to claim 1, comprising causing the apparatus to use a first and a second weight in the weighted average chosen to interpolate the sub-pixel value for a sub-pixel having co-ordinates with odd values of K and L, wherein when the chosen weighted average is that involving the value of a nearest neighbouring pixel and the value of the sub-pixel situated at co-ordinates $\frac{1}{2}$, $\frac{1}{2}$, causing the apparatus to select the relative magnitudes of the first and second weights to be inversely proportional to the respective straight-line diagonal distances of the nearest-neighbouring pixel and the sub-pixel situated at co-ordinates $\frac{1}{2}$, $\frac{1}{2}$ to the sub-pixel having co-ordinates with odd values of K and L whose value is being interpolated.

3. A method according to claim 2, comprising causing the apparatus to use first and second weights with equal values when the nearest-neighbouring pixel and sub-pixel pixel situated at co-ordinates $\frac{1}{2}$, $\frac{1}{2}$ are equidistant from the sub-pixel having co-ordinates with odd values of K and L whose value is being interpolated.

4. A method according to claim 1, comprising causing the apparatus to use a first and a second weight in the weighted average chosen to interpolate the sub-pixel value for a sub-pixel having co-ordinates with odd values of K and L, wherein when the chosen weighted average is that involving the values of a pair of diagonally-opposed sub-pixels having co-ordinates with even values of K and L, causing the apparatus to select the relative magnitudes of the first and second weights to be inversely proportional to the respective straight-line diagonal distances of the sub-pixels having co-ordinates with even values of K and L to the sub-pixel having co-ordinates with odd values of K and L whose value is being interpolated.

5. A method according to claim 4, comprising causing the apparatus to use first and second weights with equal values when the sub-pixels having co-ordinates with even values of K and L are equidistant from the sub-pixel having co-ordinates with odd values of K and L whose value is being interpolated.

6. A method according to claim 1, comprising causing the apparatus to interpolate sub-pixel values for sub-pixels having co-ordinates with even values of K and L, used in interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of K and L, using a weighted sum of the values of sub-pixels having co-ordinates with K equal to an even value and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions when the value of a sub-pixel having co-ordinates with K equal to an even value and L equal to an odd value is also required.

7. A method according to claim 1, comprising causing the apparatus to interpolate sub-pixel values for sub-pixels having co-ordinates with even values of K and L, used in interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of K and L, using a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions when the value of a sub-pixel having co-ordinates with K equal to an odd value and L equal to an even value is also required.

8. A method according to claim 1, comprising causing the apparatus to interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an odd value O and L equal to an even value E, not including zero or 2^N , by taking an average of the value of a first sub-pixel having co-ordinates with K equal to the even value O-1 and L equal to E and a second sub-pixel having co-ordinates with K equal to the even value O+1 and L equal to E.

9. A method according to claim 1, comprising causing the apparatus to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 1 and L equal to zero, by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to zero, and a sub-pixel having co-ordinates with K equal to 2 and L equal to zero.

10. A method according to claim 1, comprising causing the apparatus to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N-1 and L equal to zero, by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to zero, and a sub-pixel having co-ordinates with K equal to 2^N-2 and L equal to zero.

11. A method according to claim 1, comprising causing the apparatus to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 1 and L equal to 2^N , by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to 2 and L equal to 2^N .

53

12. A method according to claim 1, comprising causing the apparatus to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N-1 and L equal to 2^N , by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to 2^N-2 and L equal to 2^N .

13. A method according to claim 1, comprising causing the apparatus to interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an even value E, not including zero or 2^N , and L equal to an odd value O, by taking an average of the value of a first sub-pixel having co-ordinates with K equal to E and L equal to the even value O-1 and a second sub-pixel having co-ordinates with K equal to E and L equal to the even value O+1.

14. A method according to claim 1, comprising causing the apparatus to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to zero and L equal to 1, by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to zero, and a sub-pixel having co-ordinates with K equal to zero and L equal to 2.

15. A method according to claim 1, comprising causing the apparatus to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N and L equal to 1, by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to zero, and a sub-pixel having co-ordinates with K equal to 2^N and L equal to 2.

16. A method according to claim 1, comprising causing the apparatus to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to zero and L equal to 2^N-1 , by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to zero and L equal to 2^N-2 .

17. A method according to claim 1, comprising causing the apparatus to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N and L equal to 2^N-1 , by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to 2^N and L equal to 2^N-2 .

18. A method according to claim 1, comprising causing the apparatus to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N-1 and L equal to 2^N-1 , by taking a weighted average of the values of the four corner pixels that define the rectangular bounded region.

19. A method according to claim 1, in which N is one of the values 2, 3, and 4.

20. A method according to claim 1, wherein the pixels have a predetermined dynamic range, said predetermined dynamic range corresponding to the range of values said pixels can take.

21. A method according to claim 20, comprising interpolating sub-pixel values for sub-pixels having co-ordinates with K equal to an even value and L equal to zero and for sub-pixels having co-ordinates with K equal to zero and L equal to an even value, using weighted sums of the values of pixels located in rows and columns respectively, to produce a first intermediate value, said first intermediate value having a dynamic range equal to the predetermined dynamic range of the pixels used in said weighted sum multiplied by a value equal to the sum of the respective weights used in said weighted sum.

22. A method according to claim 21, wherein the dynamic range of said first intermediate value is defined by the number of bits used to represent the respective pixels used in said weighted sum plus the number of bits required to represent the sum of the respective weights used in said weighted sum.

23. A method according to claim 21, comprising truncating the first intermediate value by mathematically dividing the

54

first intermediate value by a first scale factor, said first scale factor having a value equal to the sum of the respective weights used in said weighted sum, thereby forming a sub-pixel value with a dynamic range equal to the predetermined dynamic range of said pixels.

24. A method according to claim 23, comprising using a truncated first intermediate sub-pixel value when interpolating a value for a sub-pixel having co-ordinates with odd values of K and L.

25. A method according to claim 21, comprising interpolating sub-pixel values for sub-pixels having co-ordinates with even values of K and L, using a weighted sum comprising first intermediate values for sub-pixels having co-ordinates with K equal to an even value and L equal to zero and corresponding first intermediate values for sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, to produce a second intermediate value, said second intermediate value having a dynamic range equal to the dynamic range of the first intermediate values used in said weighted sum, multiplied by a value equal to the sum of the respective weights used in said weighted sum.

26. A method according to claim 25, comprising truncating the second intermediate value by mathematically dividing the second intermediate value by a second scale factor, said second scale factor having a value equal to the sum of the respective weights used in the weighted sum of claim 21 multiplied by the sum of the respective weights used in the weighted sum of claim 25, thereby forming a sub-pixel value with a dynamic range equal to the predetermined dynamic range of said pixels.

27. A method according to claim 26, comprising using a truncated second intermediate sub-pixel value when interpolating a value for a sub-pixel having co-ordinates with odd values of K and L.

28. A method according to claim 21, comprising interpolating sub-pixel values for sub-pixels having co-ordinates with even values of K and L, using a weighted sum comprising first intermediate values for sub-pixels having co-ordinates with K equal to zero and L equal to an even value and corresponding first intermediate values for sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions to produce a second intermediate value, said second intermediate value having a dynamic range equal to the dynamic range of the first intermediate values used in said weighted sum, multiplied by a value equal to the sum of the respective weights used in said weighted sum.

29. A method according to claim 28, comprising truncating the second intermediate value by mathematically dividing the second intermediate value by a second scale factor, said second scale factor having a value equal to the sum of the respective weights used in the weighted sum of claim 21 multiplied by the sum of the respective weights used in the weighted sum of claim 28, thereby forming a sub-pixel value with a dynamic range equal to the dynamic range of said pixels.

30. A method according to claim 29, comprising using a truncated second intermediate sub-pixel value when interpolating a value for a sub-pixel having co-ordinates with odd values of K and L.

31. A method for quarter resolution sub-pixel value interpolation to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being configured for display in rows and columns, pixel and sub-pixel locations in the rows and columns being representable mathematically within the rectangular bounded region using the co-ordinate notation K/4, L/4, K and L being

55

positive integers having respective values between zero and 4, the method comprising causing an apparatus to:

interpolate a sub-pixel value for a sub-pixel having co-ordinates with K and L equal to 1 or 3, according to a predetermined choice of either a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates $\frac{3}{4}$, $\frac{3}{4}$, or a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with K and L equal to zero, 2 or 4, situated within a quadrant of the rectangular bounded region, the quadrant being defined by the sub-pixel having co-ordinates $\frac{3}{4}$, $\frac{3}{4}$ and the nearest neighbouring pixel;

interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to 2 and L equal to zero and sub-pixels having co-ordinates with K equal to zero and L equal to 2, used in the interpolation of the sub-pixels having co-ordinates with K and L equal to 1 or 3, using weighted sums of the values of pixels located in rows and columns respectively; and

interpolate a sub-pixel value for the sub-pixel having co-ordinates with both K and L equal to 2, used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with K and L equal to 1 or 3, using a predetermined choice of either a weighted sum of the values of the sub-pixel having co-ordinates with K equal to 2 and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, or a weighted sum of the value of the sub-pixel having co-ordinates with K equal to zero and L equal to 2 and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions.

32. A method for eighth resolution sub-pixel value interpolation to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being configured for display in rows and columns, pixel and sub-pixel locations in the rows and columns being representable mathematically with the rectangular bounded region using the co-ordinate notation $K/8$, $L/8$, K and L being positive integers having respective values between zero and 8, the method comprising causing an apparatus to:

interpolate a sub-pixel value for a sub-pixel having co-ordinates with K and L equal to 1, 3, 5 or 7, according to a predetermined choice of either a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates $\frac{4}{8}$, $\frac{4}{8}$, or a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with K and L equal to zero, 2, 4, 6 or 8, situated within a quadrant of the rectangular bounded region, the quadrant being defined by the sub-pixel having co-ordinates $\frac{4}{8}$, $\frac{4}{8}$ and the nearest neighbouring pixel;

interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to 2, 4 or 6 and L equal to zero and sub-pixels having co-ordinates with K equal to zero and L equal to 2, 4 or 6, used in the interpolation of the sub-pixels having co-ordinates with K and L equal to 1, 3, 5 or 7, using weighted sums of the values of pixels located in rows and columns respectively; and

interpolate sub-pixel values for sub-pixels having co-ordinates with K and L equal to 2, 4 or 6, used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with K and L equal to 1, 3, 5 or 7, using a predetermined choice of either a weighted sum of the values of sub-pixels having co-ordinates with K equal to

56

2, 4 or 6 and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions or a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to 2, 4 or 6 and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions.

33. An interpolator for sub-pixel value interpolation, the interpolator being configured to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being configured for display in rows and columns, pixel and sub-pixel locations in the rows and columns being representable mathematically within the rectangular bounded region using the co-ordinate notation $K/2^N$, $L/2^N$, K and L being positive integers having respective values between zero and 2^N , N being a positive integer greater than one and representing a particular degree of sub-pixel value interpolation, the interpolator being configured to:

interpolate a sub-pixel value for a sub-pixel having co-ordinates with odd values of K and L, according to a predetermined choice of either a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates $\frac{1}{2}$, $\frac{1}{2}$, or a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with even values of K and L, including zero, situated within a quadrant of the rectangular bounded region, the quadrant being defined by the sub-pixel having co-ordinates $\frac{1}{2}$, $\frac{1}{2}$ and the nearest neighbouring pixel;

interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an even value and L equal to zero and sub-pixels having co-ordinates with K equal to zero and L equal to an even value, used in the interpolation of the sub-pixels having co-ordinates with odd values of K and L, using weighted sums of the values of pixels located in rows and columns respectively; and

interpolate sub-pixel values for sub-pixels having co-ordinates with even values of K and L, used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of K and L, using a predetermined choice of either a weighted sum of the values of sub-pixels having co-ordinates with K equal to an even value and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, or a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions.

34. An interpolator according to claim 33, wherein the interpolator is configured to use a first and a second weight in the weighted average chosen to interpolate the sub-pixel value for a sub-pixel having co-ordinates with odd values of K and L, wherein when the chosen weighted average is that involving the value of a nearest neighbouring pixel and the value of the sub-pixel situated at co-ordinates $\frac{1}{2}$, $\frac{1}{2}$, the interpolator is configured to select the relative magnitudes of the first and second weights to be inversely proportional to the respective straight-line diagonal distances of the nearest-neighbouring pixel and the sub-pixel situated at co-ordinates $\frac{1}{2}$, $\frac{1}{2}$ to the sub-pixel having co-ordinates with odd values of K and L whose value is being interpolated.

35. An interpolator according to claim 34, wherein the interpolator is configured to use first and second weights with equal values when the nearest-neighbouring pixel and sub-

57

pixel situated at co-ordinates $\frac{1}{2}$, $\frac{1}{2}$ are equidistant from the sub-pixel having co-ordinates with odd values of K and L whose value is being interpolated.

36. An interpolator according to claim 33, wherein the interpolator is configured to use a first and a second weight in the weighted average chosen to interpolate the sub-pixel value for a sub-pixel having co-ordinates with odd values of K and L, wherein when the chosen weighted average is that involving the values of a pair of diagonally-opposed sub-pixels having co-ordinates with even values of K and L, the interpolator is configured to select the relative magnitudes of the first and second weights to be inversely proportional to the respective straight-line diagonal distances of the sub-pixels having co-ordinates with even values of K and L to the sub-pixel having co-ordinates with odd values of K and L whose value is being interpolated.

37. An interpolator according to claim 36, wherein the interpolator is configured to use first and second weights with equal values when the sub-pixels having co-ordinates with even values of K and L are equidistant from the sub-pixel having co-ordinates with odd values of K and L whose value is being interpolated.

38. An interpolator according to claim 33, wherein the interpolator is configured to interpolate sub-pixel values for sub-pixels having co-ordinates with even values of K and L, used in interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of K and L, using a weighted sum of the values of sub-pixels having co-ordinates with K equal to an even value and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions when the value of a sub-pixel having co-ordinates with K equal to an even value and L equal to an odd value is also required.

39. An interpolator according to claim 33, wherein the interpolator is configured to interpolate sub-pixel values for sub-pixels having co-ordinates with even values of K and L, used in interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of K and L, using a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions when the value of a sub-pixel having co-ordinates with K equal to an odd value and L equal to an even value is also required.

40. A method according to claim 33, comprising truncating the second intermediate value by mathematically dividing the second intermediate value by a second scale factor, said second scale factor having a value equal to the sum of the respective weights used in the weighted sum of claim 63 multiplied by the sum of the respective weights used in the weighted sum of claim 67, thereby forming a sub-pixel value with a dynamic range equal to the dynamic range of said pixels.

41. An interpolator according to claim 33, wherein the interpolator is configured to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 1 and L equal to zero, by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to zero, and a sub-pixel having co-ordinates with K equal to 2 and L equal to zero.

42. An interpolator according to claim 33, wherein the interpolator is configured to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N-1 and L equal to zero, by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to zero, and a sub-pixel having co-ordinates with K equal to 2^N-2 and L equal to zero.

58

43. An interpolator according to claim 33, wherein the interpolator is configured to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 1 and L equal to 2^N , by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to 2 and L equal to 2^N .

44. An interpolator according to claim 33, wherein the interpolator is configured to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N-1 and L equal to 2^N , by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to 2^N-2 and L equal to 2^N .

45. An interpolator according to claim 33, wherein the interpolator is configured to interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an even value E, not including zero or 2^N , and L equal to an odd value O, by taking an average of the value of a first sub-pixel having co-ordinates with K equal to E and L equal to the even value O-1 and a second sub-pixel having co-ordinates with K equal to E and L equal to the even value O+1.

46. An interpolator according to claim 33, wherein the interpolator is configured to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to zero and L equal to 1, by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to zero, and a sub-pixel having co-ordinates with K equal to zero and L equal to 2.

47. An interpolator according to claim 33, wherein the interpolator is configured to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N and L equal to 1, by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to zero, and a sub-pixel having co-ordinates with K equal to 2^N and L equal to 2.

48. An interpolator according to claim 33, wherein the interpolator is configured to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to zero and L equal to 2^N-1 , by taking an average of the value of a pixel having co-ordinates with K equal to zero and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to zero and L equal to 2^N-2 .

49. An interpolator according to claim 33, wherein the interpolator is configured to interpolate a sub-pixel value for a sub-pixel having co-ordinates with K equal to 2^N and L equal to 2^N-1 , by taking an average of the value of a pixel having co-ordinates with K equal to 2^N and L equal to 2^N , and a sub-pixel having co-ordinates with K equal to 2^N and L equal to 2^N-2 .

50. An interpolator according to claim 33, wherein the interpolator is configured to interpolate a sub-pixel value for the sub-pixel having co-ordinates with K equal to 2^N-1 and L equal to 2^N-1 , by taking a weighted average of the values of the four corner pixels that define the rectangular bounded region.

51. An interpolator according to claim 33, wherein N is set equal to 2.

52. An interpolator according to claim 33, wherein N is set equal to 3.

53. An interpolator according to claim 33, wherein the pixels have a predetermined dynamic range, said predetermined dynamic range corresponding to the range of values said pixels can take.

54. An interpolator according to claim 53, wherein the interpolator is configured to interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an even value

59

and L equal zero and for sub-pixels having co-ordinates with K equal to zero and L equal to an even value, using weighted sums of the values of pixels located in rows and columns respectively, to produce a first intermediate value, said first intermediate value having a dynamic range equal to the predetermined dynamic range of the pixels used in said weighted sum multiplied by a value equal to the sum of the respective weights used in said weighted sum.

55. An interpolator according to claim 54, wherein the dynamic range of said first intermediate value is defined by the number of bits used to represent the respective pixels used in said weighted sum plus the number of bits required to represent the sum of the respective weights used in said weighted sum.

56. An interpolator according to claim 54, wherein the interpolator is configured to truncate the first intermediate value by mathematically dividing the first intermediate value by a first scale factor, said first scale factor having a value equal to the sum of the respective weights used in said weighted sum, thereby forming a sub-pixel value with a dynamic range equal to the predetermined dynamic range of said pixels.

57. An interpolator according to claim 56, wherein the interpolator is configured to use a truncated first intermediate sub-pixel value when interpolating a value for a sub-pixel having co-ordinates with odd values of K and L.

58. An interpolator according to claim 54, wherein the interpolator is configured to interpolate sub-pixel values for sub-pixels having co-ordinates with even values of K and L, using a weighted sum comprising first intermediate values for sub-pixels having co-ordinates with K equal to an even value and L equal to zero and corresponding first intermediate values for sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, to produce a second intermediate value, said second intermediate value having a dynamic range equal to the dynamic range of the first intermediate values used in said weighted sum, multiplied by a value equal to the sum of the respective weights used in said weighted sum.

59. An interpolator according to claim 58, wherein the interpolator is configured to truncate the second intermediate value by mathematically dividing the second intermediate value by a second scale factor, said second scale factor having a value equal to the sum of the respective weights used in the weighted sum of claim 54 multiplied by the sum of the respective weights used in the weighted sum of claim 58, thereby forming a sub-pixel value with a dynamic range equal to the predetermined dynamic range of said pixels.

60. An interpolator according to claim 59, wherein the interpolator is configured to use a truncated second intermediate sub-pixel value when interpolating a value for a sub-pixel having co-ordinates with odd values of K and L.

61. An interpolator according to claim 54, wherein the interpolator is configured to interpolate sub-pixel values for sub-pixels having co-ordinates with even values of K and L, using a weighted sum comprising first intermediate values for sub-pixels having co-ordinates with K equal to zero and L equal to an even value and corresponding first intermediate values for sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded region to produce a second intermediate value, said second intermediate value having a dynamic range equal to the dynamic range of the first intermediate values used in said weighted sum, multiplied by a value equal to the sum of the respective weights used in said weighted sum.

62. An interpolator according to claim 61, wherein the interpolator is configured to truncate the second intermediate

60

ate value by mathematically dividing the second intermediate value by a second scale factor, said second scale factor having a value equal to the sum of the respective weights used in the weighted sum of claim 54 multiplied by the sum of the respective weights used in the weighted sum of claim 61, thereby forming a sub-pixel value with a dynamic range equal to the predetermined dynamic range of said pixels.

63. An interpolator according to claim 62, wherein the interpolator is configured to use a truncated second intermediate sub-pixel value when interpolating a value for a sub-pixel having co-ordinates with odd values of K and L.

64. A video encoder comprising an interpolator for sub-pixel value interpolation, the interpolator being configured to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being configured for display in rows and columns, pixel and sub-pixel locations in the rows and columns being representable mathematically within the rectangular bounded region using the co-ordinate notation $K/2^N$, $L/2^N$, K and L being positive integers having respective values between zero and 2^N , N being a positive integer greater than one and representing a particular degree of sub-pixel value interpolation, the interpolator being configured to:

interpolate a sub-pixel value for a sub-pixel having co-ordinates with odd values of K and L, according to a predetermined choice of either a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates $\frac{1}{2}$, $\frac{1}{2}$, or a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with even values of K and L, including zero, situated within a quadrant of the rectangular bounded region, the quadrant being defined by the sub-pixel having co-ordinates $\frac{1}{2}$, $\frac{1}{2}$ and the nearest neighbouring pixel;

interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an even value and L equal to zero and sub-pixels having co-ordinates with K equal to zero and L equal to an even value, used in the interpolation of the sub-pixels having co-ordinates with odd values of K and L, using weighted sums of the values of pixels located in rows and columns respectively; and

interpolate sub-pixel values for sub-pixels having co-ordinates with even values of K and L, used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of K and L, using a predetermined choice of either a weighted sum of the values of sub-pixels having co-ordinates with K equal to an even value and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, or a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent bounded rectangular regions.

65. A codec comprising a video encoder according to claim 64 and a video decoder comprising an interpolator for sub-pixel value interpolation, the interpolator being configured to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being configured for display in rows and columns, pixel and sub-pixel locations in the rows and columns being representable mathematically within the rectangular bounded region using the co-ordinate notation $K/2^N$, $L/2^N$, K and L being positive integers having respective values between zero and

61

2^N , N being a positive integer greater than one and representing a particular degree of sub-pixel value interpolation, the interpolator being configured to:

interpolate a sub-pixel value for a sub-pixel having co-ordinates with odd values of K and L, according to a predetermined choice of either a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates $\frac{1}{2}$, $\frac{1}{2}$, or a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with even values of K and L, including zero, situated within a quadrant of the rectangular bounded region, the quadrant being defined by the sub-pixel having co-ordinates $\frac{1}{2}$, $\frac{1}{2}$ and the nearest neighbouring pixel;

interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an even value and L equal to zero and sub-pixels having co-ordinates with K equal to zero and L equal to an even value, used in the interpolation of the sub-pixels having co-ordinates with odd values of K and L, using weighted sums of the values of pixels located in rows and columns respectively; and

interpolate sub-pixel values for sub-pixels having co-ordinates with even values of K and L, used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of K and L, using a predetermined choice of either a weighted sum of the values of sub-pixels having co-ordinates with K equal to an even value and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, or a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions.

66. A communications terminal comprising a video encoder according to claim 64.

67. A communications terminal according to claim 66, wherein the communications terminal is a wireless terminal.

68. A still image encoder comprising an interpolator for sub-pixel value interpolation, the interpolator being configured to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being configured for display in rows and columns, pixel and sub-pixel locations in the rows and columns being representable mathematically within the rectangular bounded region using the co-ordinate notation $K/2^N$, $L/2^N$, K and L being positive integers having respective values between zero and 2^N , N being a positive integer greater than one and representing a particular degree of sub-pixel value interpolation, the interpolator being configured to:

interpolate a sub-pixel value for a sub-pixel having co-ordinates with odd values of both K and L, according to a predetermined choice of either a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates $\frac{1}{2}$, $\frac{1}{2}$, or a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with even values of K and L, including zero, situated within a quadrant of the rectangular bounded region, the quadrant being defined by the sub-pixel having co-ordinates $\frac{1}{2}$, $\frac{1}{2}$ and the nearest neighbouring pixel;

interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an even value and L equal to zero and sub-pixels having co-ordinates with K equal to zero and L equal to an even value, used in the interpolation of

62

the sub-pixels having co-ordinates with odd values of K and L, using weighted sums of the values of pixels located in rows and columns respectively; and

interpolate sub-pixel values for sub-pixels having co-ordinates with even values of K and L, used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of K and L, using a predetermined choice of either a weighted sum of the values of sub-pixels having co-ordinates with K equal to an even value and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, or a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions.

69. A codec comprising a still image encoder according to claim 68 and a still image decoder comprising an interpolator for sub-pixel value interpolation, the interpolator being configured to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being configured for display in rows and columns, pixel and sub-pixel locations in the rows and columns being representable mathematically within the rectangular bounded region using the co-ordinate notation $K/2^N$, $L/2^N$, K and L being positive integers having respective values between zero and 2^N , N being a positive integer greater than one and representing a particular degree of sub-pixel value interpolation, the interpolator being configured to:

interpolate a sub-pixel value for a sub-pixel having co-ordinates with odd values of K and L, according to a predetermined choice of either a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates $\frac{1}{2}$, $\frac{1}{2}$, or a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with even values of K and L, including zero, situated within a quadrant of the rectangular bounded region, the quadrant being defined by the sub-pixel having co-ordinates $\frac{1}{2}$, $\frac{1}{2}$ and the nearest neighbouring pixel;

interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an even value and L equal to zero and sub-pixels having co-ordinates with K equal to zero and L equal to an even value, used in the interpolation of the sub-pixels having co-ordinates with odd values of K and L, using weighted sums of the values of pixels located in rows and columns respectively; and

interpolate sub-pixel values for sub-pixels having co-ordinates with even values of K and L, used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of K and L, using a predetermined choice of either a weighted sum of the values of sub-pixels having co-ordinates with K equal to an even value and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, or a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions.

70. A communications terminal comprising a still image encoder according to claim 68.

71. A video encoder comprising an interpolator for sub-pixel value interpolation, the interpolator being configured to

63

determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being configured for display in rows and columns, pixel and sub-pixel locations in the rows and columns being representable mathematically within the rectangular bounded region using the co-ordinate notation $K/2^N$, $L/2^N$, K and L being positive integers having respective values between zero and 2^N , N being a positive integer greater than one and representing a particular degree of sub-pixel value interpolation, the interpolator being configured to:

interpolate a sub-pixel value for a sub-pixel having co-ordinates with odd values of K and L, according to a predetermined choice of either a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates $\frac{1}{2}$, $\frac{1}{2}$, or a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with even values of K and L, including zero, situated within a quadrant of the rectangular bounded region, the quadrant being defined by the sub-pixel having co-ordinates $\frac{1}{2}$, $\frac{1}{2}$ and the nearest neighbouring pixel;

interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an even value and L equal to zero and sub-pixels having co-ordinates with K equal to zero and L equal to an even value, used in the interpolation of the sub-pixels having co-ordinates with odd values of K and L, using weighted sums of the values of pixels located in rows and columns respectively; and

interpolate sub-pixel values for sub-pixels having co-ordinates with even values of K and L, used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of K and L, using a predetermined choice of either a weighted sum of the values of sub-pixels having co-ordinates with K equal to an even value and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, or a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded region.

72. A communications terminal comprising a video decoder according to claim 71.

73. A still image decoder comprising an interpolator for sub-pixel value interpolation, the interpolator being configured to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being configured for display in rows and columns, pixel and sub-pixel locations in the rows and columns being representable mathematically within the rectangular bounded region using the co-ordinate notation $K/2^N$, $L/2^N$, K and L being positive integers having respective values between zero and 2^N , N being a positive integer greater than one and representing a particular degree of sub-pixel value interpolation, the interpolator being configured to:

interpolate a sub-pixel value for a sub-pixel having co-ordinates with odd values of K and L, according to a predetermined choice of either a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates $\frac{1}{2}$, $\frac{1}{2}$, or a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with even values of K and L, including zero, situated within a quadrant of the rectangular bounded region, the quadrant

64

being defined by the sub-pixel having co-ordinates $\frac{1}{2}$, $\frac{1}{2}$ and the nearest neighbouring pixel;

interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an even value and L equal to zero and sub-pixels having co-ordinates with K equal to zero and L equal to an even value, used in the interpolation of the sub-pixels having co-ordinates with odd values of K and L, using weighted sums of the values of pixels located in rows and columns respectively; and

interpolate sub-pixel values for sub-pixels having co-ordinates with even values of K and L, used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of K and L, using a predetermined choice of either a weighted sum of the values of sub-pixels having co-ordinates with K equal to an even value and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, or a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions.

74. A communications terminal comprising a still image decoder according to claim 73.

75. An interpolator for sub-pixel value interpolation, the interpolator being configured to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being configured for display in rows and columns, pixel and sub-pixel locations in the rows and columns being representable mathematically within the rectangular bounded region using the co-ordinate notation $K/2^N$, $L/2^N$, K and L being positive integers having respective values between zero and 2^N , N being a positive integer greater than one and representing a particular degree of sub-pixel value interpolation, the interpolator comprising:

circuitry configured to interpolate a sub-pixel value for a sub-pixel having co-ordinates with odd values of K and L, according to a predetermined choice of either a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates $\frac{1}{2}$, $\frac{1}{2}$, or a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with even values of K and L, including zero, situated within a quadrant of the rectangular bounded region, the quadrant being defined by the sub-pixel having co-ordinates $\frac{1}{2}$, $\frac{1}{2}$ and the nearest neighbouring pixel;

circuitry configured to interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an even value and L equal to zero and sub-pixels having co-ordinates with K equal to zero and L equal to an even value, used in the interpolation of the sub-pixels having co-ordinates with odd values of K and L, using weighted sums of the values of pixels located in rows and columns respectively; and

circuitry configured to interpolate sub-pixel values for sub-pixels having co-ordinates with even values of K and L, used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of K and L, using a predetermined choice of either a weighted sum of the values of sub-pixels having co-ordinates with K equal to an even value and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, or a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even

65

value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions.

76. An interpolator for quarter resolution sub-pixel value interpolation, the interpolator being configured to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being configured for display in rows and columns, pixel and sub-pixel locations in the rows and columns being representable mathematically within the rectangular bounded region using the co-ordinate notation $K/4$, $L/4$, K and L being positive integers having respective values between zero and 4, the interpolator being configured to:

interpolate a sub-pixel value for a sub-pixel having co-ordinates with K and L equal to 1 or 3, according to a predetermined choice of either a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates $3/4$, $3/4$, or a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with K and L equal to zero, 2 or 4, situated within a quadrant of the rectangular bounded region, the quadrant being defined by the sub-pixel having co-ordinates $3/4$, $3/4$ and the nearest neighbouring pixel;

interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to 2 and L equal to zero and sub-pixels having co-ordinates with K equal to zero and L equal to 2, used in the interpolation of the sub-pixels having co-ordinates with K and L equal to 1 or 3, using weighted sums of the values of pixels located in rows and columns respectively; and

interpolate a sub-pixel value for the sub-pixel having co-ordinates with both K and L equal to 2, used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with both K and L equal to 1 or 3, using a predetermined choice of either a weighted sum of the values of the sub-pixel having co-ordinates with K equal to 2 and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, or a weighted sum of the value of the sub-pixel having co-ordinates with K equal to zero and L equal to 2 and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions.

77. A video encoder or still image encoder comprising an interpolator according to claim 76.

78. A communications terminal comprising a video encoder or still image encoder according to claim 77.

79. An interpolator for eighth resolution sub-pixel value interpolation, the interpolator being configured to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being configured for display in rows and columns, pixel and sub-pixel locations in the rows and columns being representable mathematically within the rectangular bounded region using the co-ordinate notation $K/8$, $L/8$, K and L being positive integers having respective values between zero and 8, the interpolator being configured to:

interpolate a sub-pixel value for a sub-pixel having co-ordinates with K and L equal to 1, 3, 5 or 7, according to a predetermined choice of either a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates $7/8$, $7/8$, or a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with both K and

66

L equal to zero, 2, 4, 6 or 8, situated within a quadrant of the rectangular bounded region, the quadrant being defined by the sub-pixel having co-ordinates $7/8$, $7/8$ and the nearest neighbouring pixel;

interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to 2, 4 or 6 and L equal to zero and sub-pixels having co-ordinates with K equal to zero and L equal to 2, 4 or 6, used in the interpolation of the sub-pixels having co-ordinates with both K and L equal to 1, 3, 5 or 7, using weighted sums of the values of pixels located in rows and columns respectively; and

interpolate sub-pixel values for sub-pixels having co-ordinates with K and L equal to 2, 4 or 6, used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with K and L equal to 1, 3, 5 or 7, using a predetermined choice of either a weighted sum of the values of sub-pixels having co-ordinates with K equal to 2, 4 or 6 and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions or a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to 2, 4 or 6 and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions.

80. A video decoder or still image decoder comprising an interpolator according to claim 79.

81. A communications terminal comprising a video decoder or still image decoder according to claim 80.

82. A computer program for sub-pixel value interpolation, embodied on a computer readable storage medium, the computer program being configured to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being configured for display in rows and columns, pixel and sub-pixel locations in the rows and columns being representable mathematically within the rectangular bounded region using the co-ordinate notation $K/2^N$, $L/2^N$, K and L being positive integers having respective values between zero and 2^N , N being a positive integer greater than one and representing a particular degree of sub-pixel value interpolation, the computer program comprising:

computer program code for interpolating a sub-pixel value for a sub-pixel having co-ordinates with odd values of both K and L , according to a predetermined choice of either a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates $1/2$, $1/2$, or a weighted average of the values of a pair of diagonally-opposed sub-pixels having co-ordinates with even values of both K and L , including zero, situated within a quadrant of the rectangular bounded region, the quadrant being defined by the sub-pixel having co-ordinates $1/2$, $1/2$ and the nearest neighbouring pixel;

computer program code for interpolating sub-pixel values for sub-pixels having co-ordinates with K equal to an even value and L equal to zero and sub-pixels having co-ordinates with K equal to zero and L equal to an even value, used in the interpolation of the sub-pixels having co-ordinates with odd values of K and L , using weighted sums of the values of pixels located in rows and columns respectively; and

computer program code for interpolating sub-pixel values for sub-pixels having co-ordinates with even values of K and L , used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of K and L , using a predetermined choice of either a weighted sum of the values of sub-pixels having co-ordinates with

67

K equal to an even value and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, or a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions.

83. A network apparatus comprising an interpolator for sub-pixel value interpolation, the interpolator being configured to determine values for sub-pixels situated within a rectangular bounded region defined by four corner pixels with no intermediate pixels between the corners, the pixels and sub-pixels being configured for display in rows and columns, pixel and sub-pixel locations in the rows and columns being representable mathematically within the rectangular bounded region using the co-ordinate notation $K/2^N$, $L/2^N$, K and L being positive integers having respective values between zero and 2^N , N being a positive integer greater than one and representing a particular degree of sub-pixel value interpolation, the interpolator being configured to:

interpolate a sub-pixel value for a sub-pixel having co-ordinates with odd values of K and L, according to a predetermined choice of either a weighted average of the value of a nearest-neighbouring pixel and the value of the sub-pixel situated at co-ordinates $\frac{1}{2}$, $\frac{1}{2}$, or a weighted average of the values of a pair of diagonally-

68

opposed sub-pixels having co-ordinates with even values of K and L, including zero, situated within a quadrant of the rectangular bounded region, the quadrant being defined by the sub-pixel having co-ordinates $\frac{1}{2}$, $\frac{1}{2}$ and the nearest neighbouring pixel;

interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an even value and L equal to zero and sub-pixels having co-ordinates with K equal to zero and L equal to an even value, used in the interpolation of the sub-pixels having co-ordinates with odd values of K and L, using weighted sums of the values of pixels located in rows and columns respectively; and

interpolate sub-pixel values for sub-pixels having co-ordinates with even values of K and L, used in the interpolation of sub-pixel values for the sub-pixels having co-ordinates with odd values of K and L, using a predetermined choice of either a weighted sum of the values of sub-pixels having co-ordinates with K equal to an even value and L equal to zero and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions, or a weighted sum of the values of sub-pixels having co-ordinates with K equal to zero and L equal to an even value and the values of sub-pixels having corresponding co-ordinates in immediately adjacent rectangular bounded regions.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 8,036,273 B2
APPLICATION NO. : 11/839205
DATED : October 11, 2011
INVENTOR(S) : Karczewicz et al.

Page 1 of 2

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Title Page, item (75); Column 1, line 2, in the Second Inventor's Name, delete "Antti Hallapuro" and insert -- Antti Olli Hallapuro -- therefor.

Column 1, line 2, in the Second Inventor's Residence, delete "FL (US)" and insert -- (FI) -- therefor.

Column 51, line 64, in Claim 3, after -- pixel -- delete "pixel".

Column 55, line 22, in Claim 31, after -- with -- delete "both".

Column 57, line 46-53, delete Claim 40 and insert -- An interpolator according to claim 33, wherein the interpolator is configured to interpolate sub-pixel values for sub-pixels having co-ordinates with K equal to an odd value O and L equal to an even value E, not including zero or 2^N , by taking an average of the value of a first sub-pixel having co-ordinates with K equal to the even value O - 1 and L equal to E and a second sub-pixel having co-ordinates with K equal to the even value O + 1 and L equal to E. -- therefor.

Column 57, line 66, in Claim 42, delete " $2^N 31 2$ " and insert -- $2^N - 2$ -- therefor.

Column 59, line 1, in Claim 54, delete "equal" and insert -- equal to -- therefor.

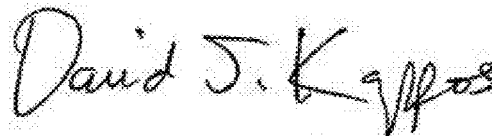
Column 59, line 60, in Claim 61, delete "region" and insert -- regions -- therefor.

Column 59, line 67, in Claim 62, delete "truncated" and insert -- truncate -- therefor.

Column 60, line 55-56, in Claim 64, delete "bounded rectangular regions" and insert -- rectangular bounded regions -- therefor.

Column 61, line 6, in Claim 65, delete "of a either" and insert -- of either a -- therefor.

Signed and Sealed this
Sixth Day of March, 2012



David J. Kappos
Director of the United States Patent and Trademark Office

CERTIFICATE OF CORRECTION (continued)

Page 2 of 2

U.S. Pat. No. 8,036,273 B2

Column 61, line 13, in Claim 65, delete “co- ordinates” and insert -- co-ordinates -- therefor.

Column 61, line 54, in Claim 68, after -- of -- delete “both”.

Column 62, line 41, in Claim 69, delete “co- ordinates” and insert -- co-ordinates -- therefor.

Column 62, line 66, in Claim 71, delete “encoder” and insert -- decoder -- therefor.

Column 63, line 43, in Claim 71, delete “region” and insert -- regions -- therefor.

Column 65, line 34, in Claim 76, after -- with -- delete “both”.

Column 65, line 36, in Claim 76, after -- with -- delete “both”.

Column 65, line 67, in Claim 79, after -- with -- delete “both”.

Column 66, line 9, in Claim 79, after -- with -- delete “both”.

Column 66, line 44, in Claim 82, after -- of -- delete “both”.

Column 66, line 49, in Claim 82, after -- of -- delete “both”.

EXHIBIT 10



US006856701B2

(12) **United States Patent**
Karczewicz et al.

(10) **Patent No.:** **US 6,856,701 B2**
(45) **Date of Patent:** **Feb. 15, 2005**

(54) **METHOD AND SYSTEM FOR CONTEXT-BASED ADAPTIVE BINARY ARITHMETIC CODING**

(75) Inventors: **Marta Karczewicz**, Irving, TX (US);
Ragip Kurceren, Irving, TX (US)

(73) Assignee: **Nokia Corporation**, Espoo (FI)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/995,240**

(22) Filed: **Nov. 27, 2001**

(65) **Prior Publication Data**

US 2003/0081850 A1 May 1, 2003

Related U.S. Application Data

(60) Provisional application No. 60/322,112, filed on Sep. 14, 2001.

(51) **Int. Cl.**⁷ **G06K 9/36**; G06K 9/34

(52) **U.S. Cl.** **382/247**; 382/173; 382/248

(58) **Field of Search** 382/173, 232,
382/233, 240, 247, 248, 250, 245, 251;
341/50, 59, 52, 67, 69, 79, 106, 107; 358/505

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,243,496 B1 6/2001 Wilkinson 382/245
6,327,392 B1 * 12/2001 Li 382/248
6,339,658 B1 * 1/2002 Moccagatta et al. 382/240
6,351,563 B1 2/2002 Kim et al. 382/232
6,545,618 B2 * 4/2003 Yip 341/107

6,567,081 B1 * 5/2003 Li et al. 345/419
6,577,251 B1 * 6/2003 Yip 341/50

OTHER PUBLICATIONS

I. H. Witten et al., "Arithmetic Coding for Data Compression," *Commun ACM*, vol. 30, pp. 520-540, Jun. 1987).

H.26L Test Model Long Term No. 8 (TML-8) draft0 ITU-T Telecommunication Standardization Section, Study Group 16, Video Coding Experts Group.

Gisle Bjontegaard, Recommended Simulation Conditions for H.26L (VCG-M75, ITU-T Video Coding Experts Group, Austin, TX USA, Apr. 2-4, 2001).

* cited by examiner

Primary Examiner—Andrew W. Johns

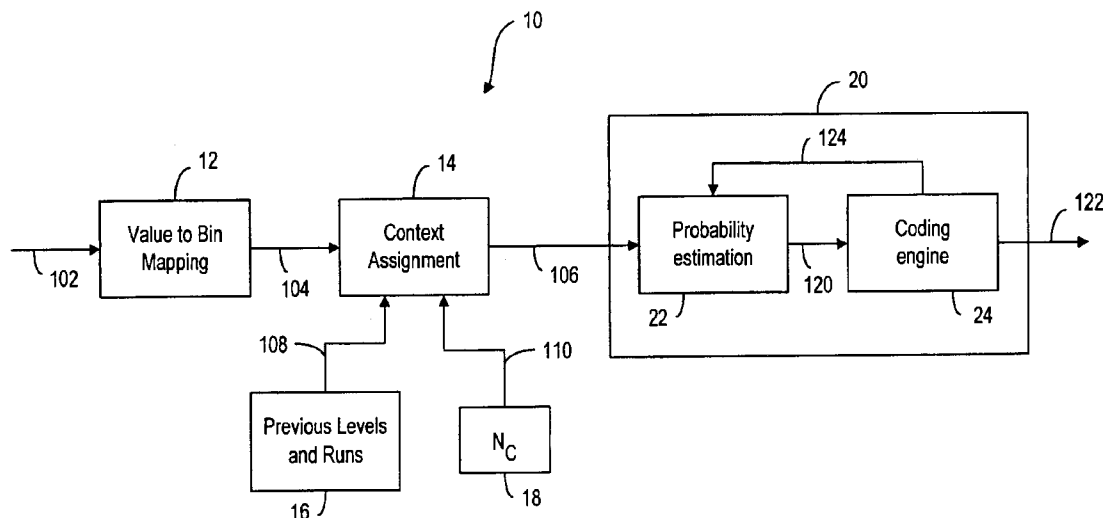
Assistant Examiner—Amir Alavi

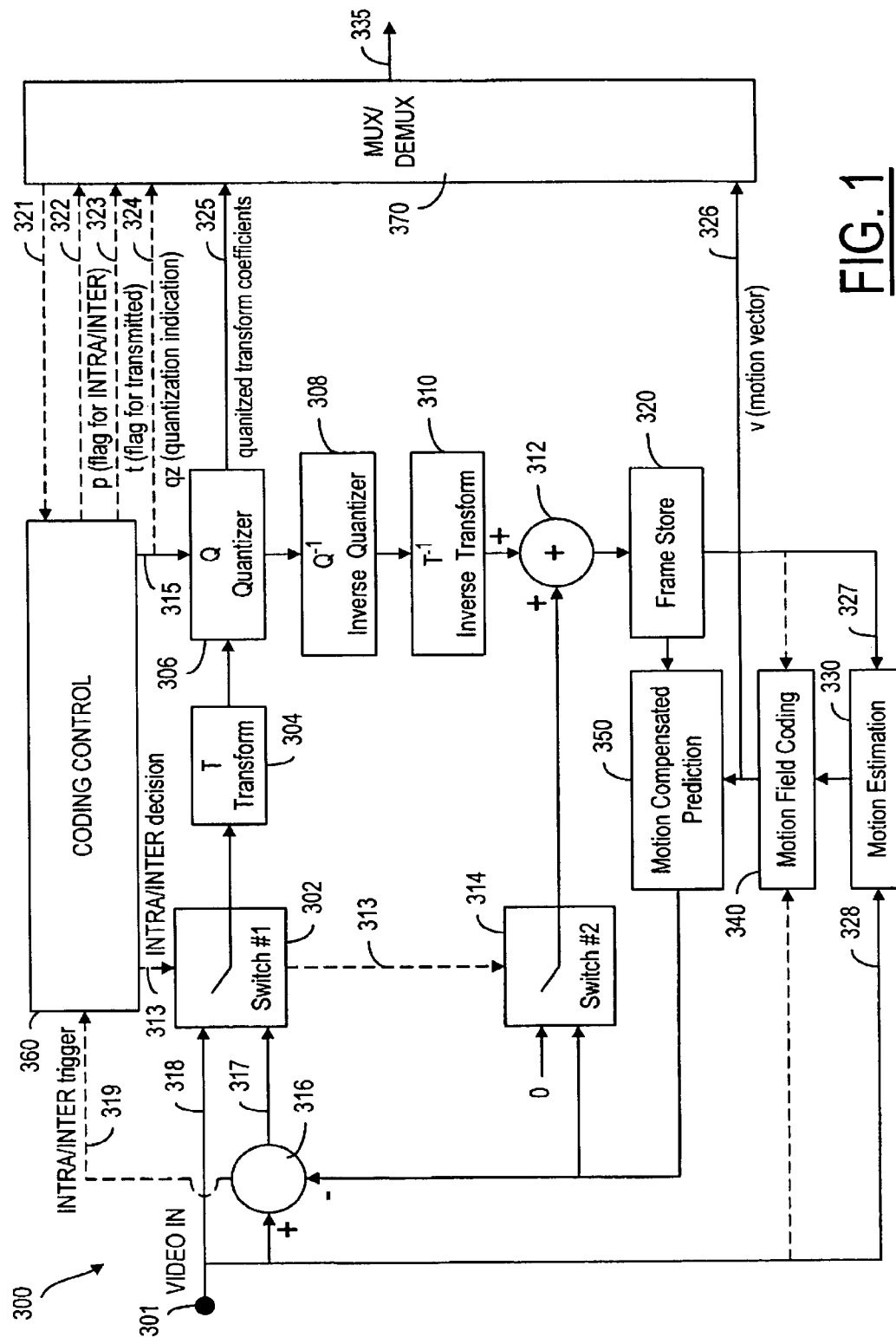
(74) *Attorney, Agent, or Firm*—Ware, Fressola, Va Der Sluys & Adolphson LLP

(57) **ABSTRACT**

A method and system for image coding, wherein an image is divided into a plurality of blocks for scanning. The pixels values in the scanned block are represented by a plurality of level-run value pairs, wherein the level value is indicative of a non-zero pixel value and the run value is indicative of the number of consecutive zero pixel values preceding the non-zero pixel value. A plurality of contexts indicative of the level-run value pairs are conveyed to a decoder for allowing the decoder to reconstruct the image based on the contexts. The assignment of the contexts is also based on the level value of a preceding level-run pair. Additionally, instead of an end-of-block symbol, the number of non-zero coefficients is provided to the decoder prior to conveying the contexts thereto.

40 Claims, 16 Drawing Sheets





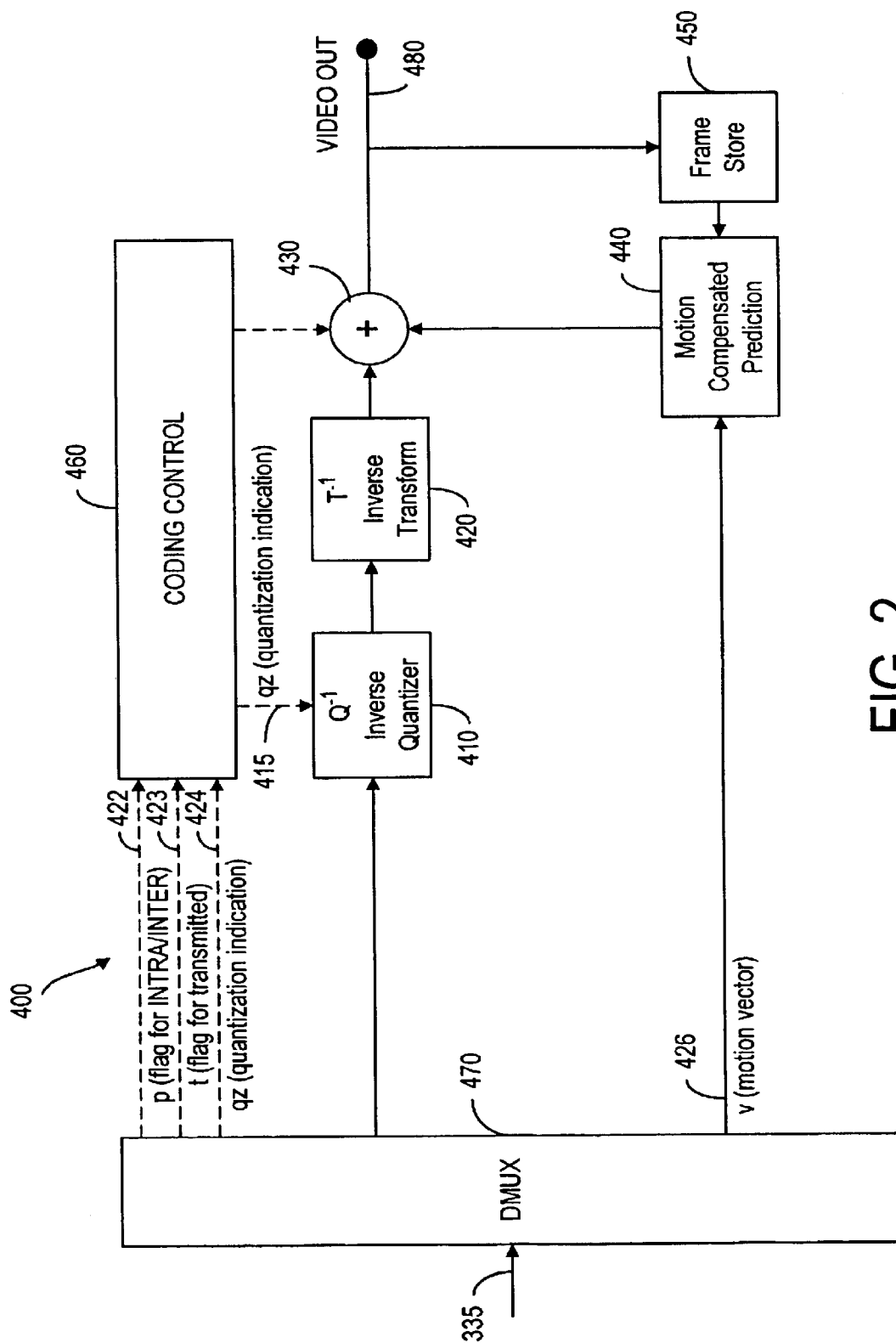


FIG. 2

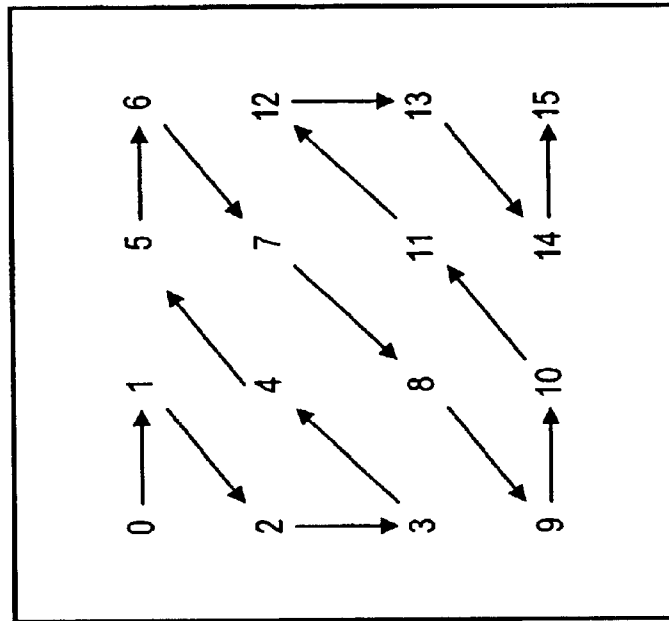


FIG. 3

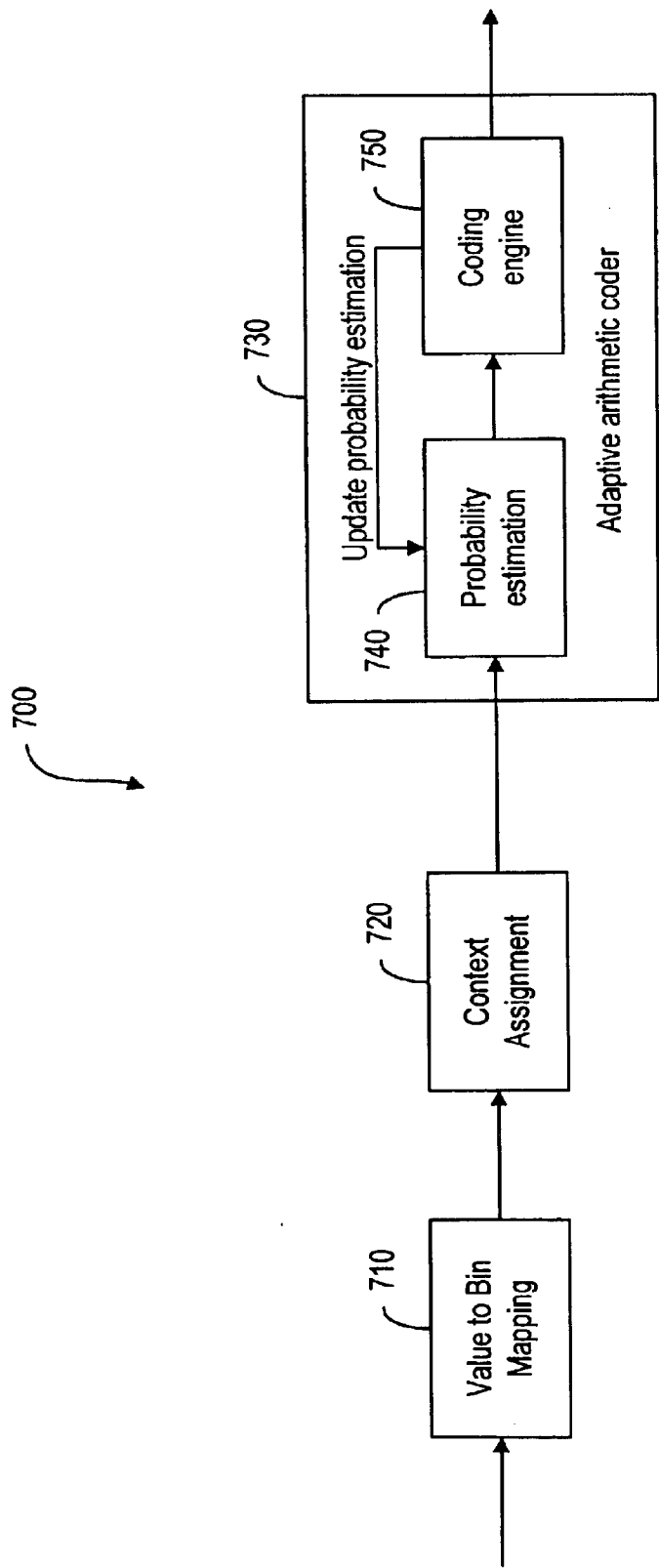


FIG. 4
(PRIOR ART)

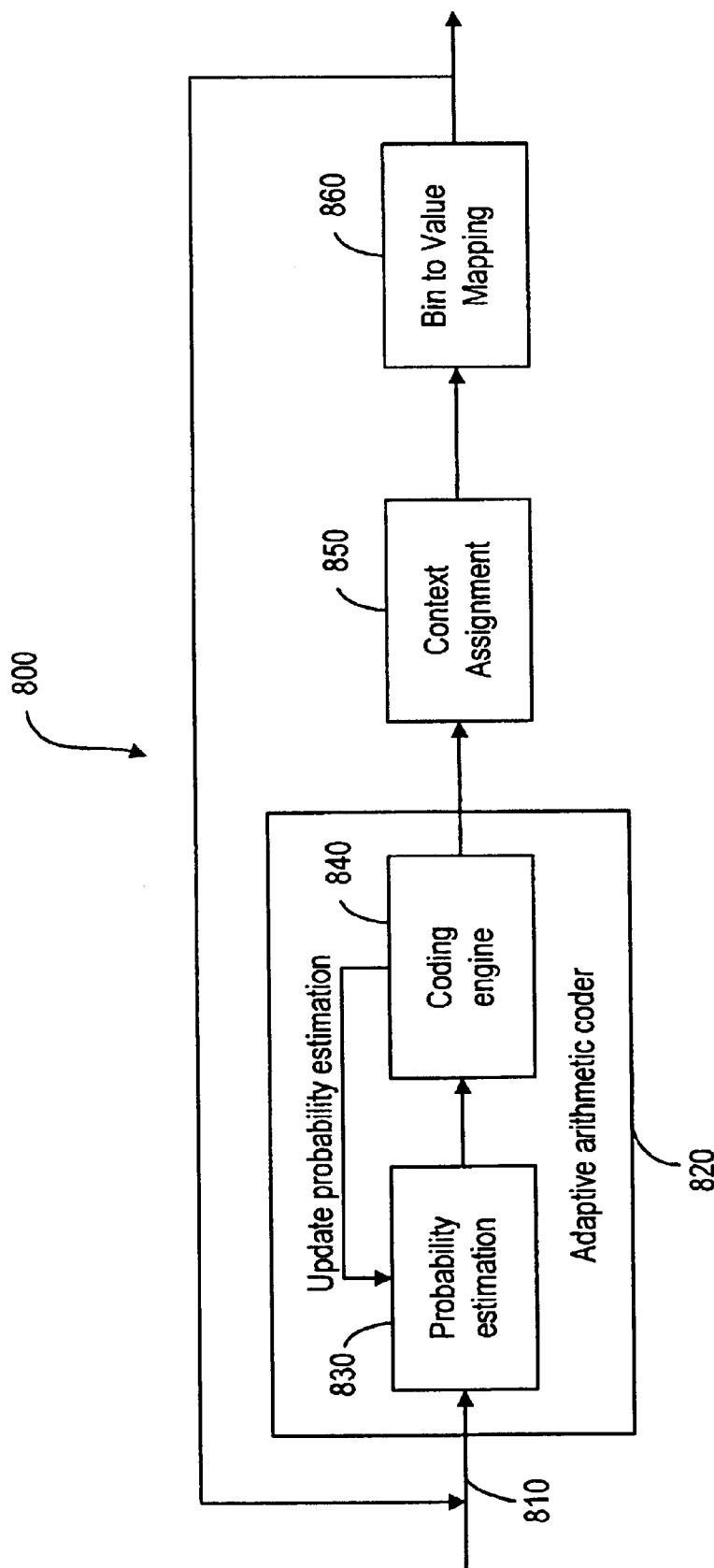


FIG. 5
(PRIOR ART)

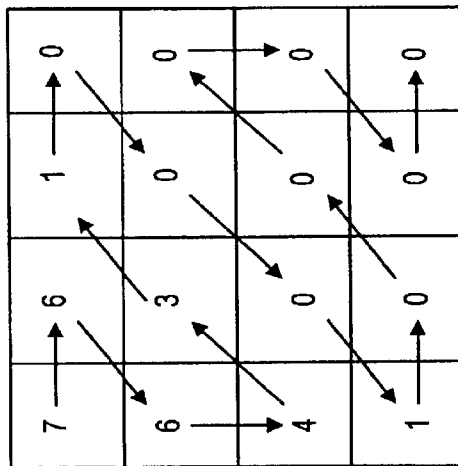


FIG. 6a

07	06	06	04	03	01	31	0
70	60	60	40	30	10	13	0

↑ Run Level

FIG. 6b

FIG. 6b

[illegible]

FIG. 6C

		Initial state			After encoding (4 x 4 block)		
		0	1	0	1	0	1
Level context	1	1	1	1	1	8	1
	2	1	1	1	1	6	3
	3	1	1	1	1	3	6
Run context	1	1	1	1	1	2	7
	2	1	1	1	1	8	1
	3	1	1	1	1	7	2

FIG. 6d

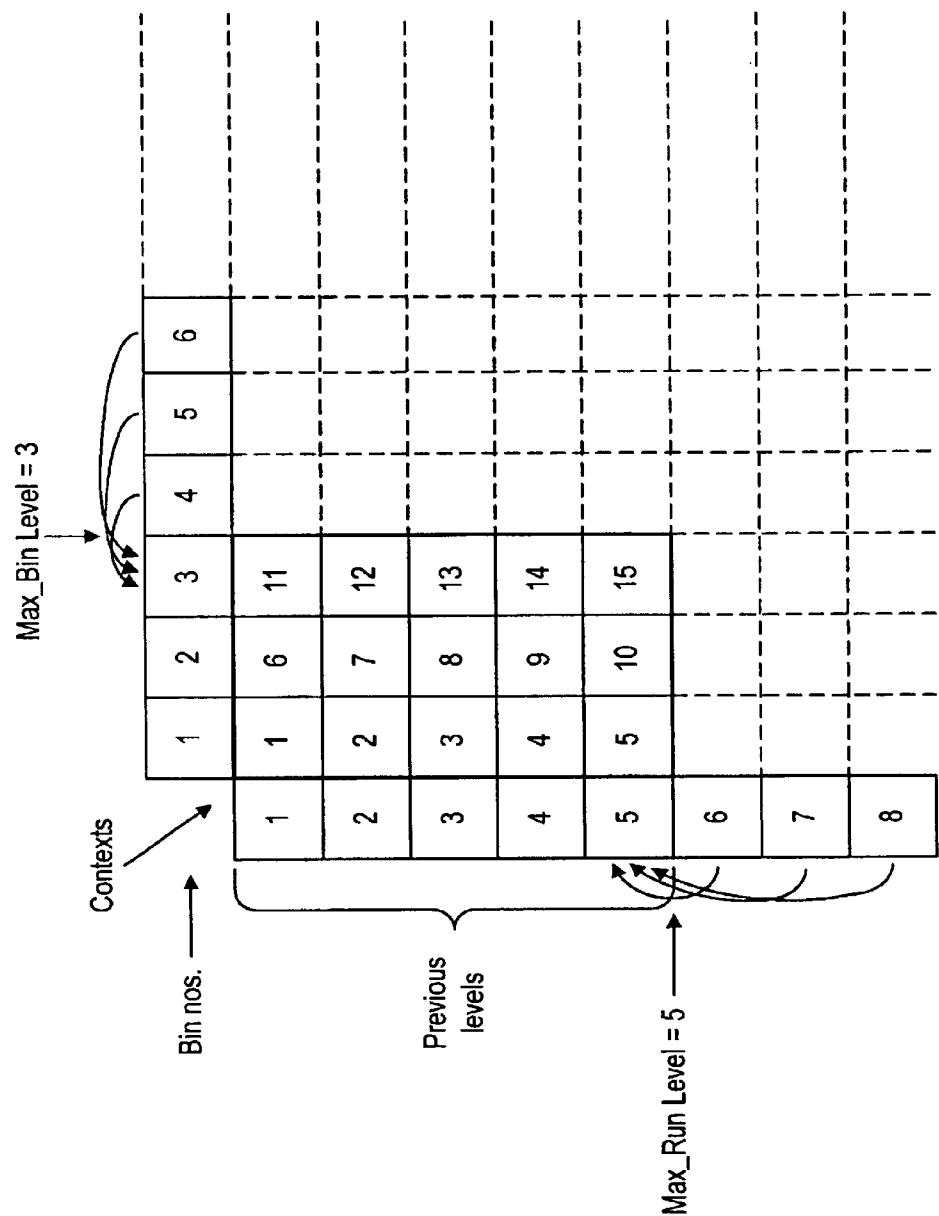


FIG. 7a

	Bin to which level is mapped	Prev_Level	Context = (bin_nr - 1) * Max_Level + Prev_Level
Max_Bin_Level = 3	1	1	1
	2	1	6
	≥ 3	1	11
Max_Level = 5	1	2	2
	2	2	7
	≥ 3	2	12
	1	3	3
	2	3	8
	≥ 3	3	13
	1	4	4
	2	4	9
	≥ 3	4	14
	1	≥ 5	5
	2	≥ 5	10
	≥ 3	≥ 5	15

Context Mapping Table

FIG. 7b

	Bin to which run is mapped	Level associated with run	Context = (bin_nr - 1) * Max_RunL + Level
Max_Bin_Run = 3	1	1	1
	2	1	5
	≥ 3	1	9
Max_RunL = 4	1	2	2
	2	2	6
	≥ 3	2	10
	1	3	3
	2	3	7
	≥ 3	3	11
	1	≥ 4	4
	2	≥ 4	8
	≥ 3	≥ 4	12

FIG. 8a

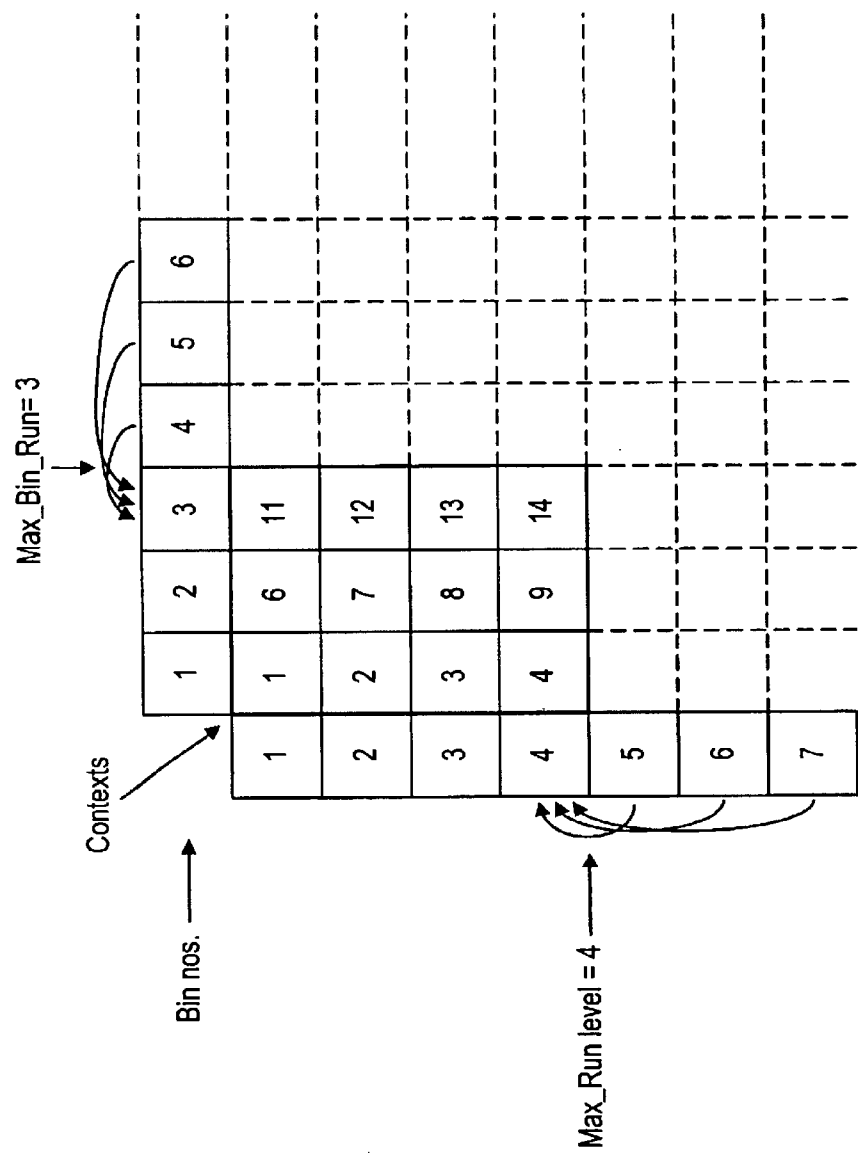


FIG. 8b

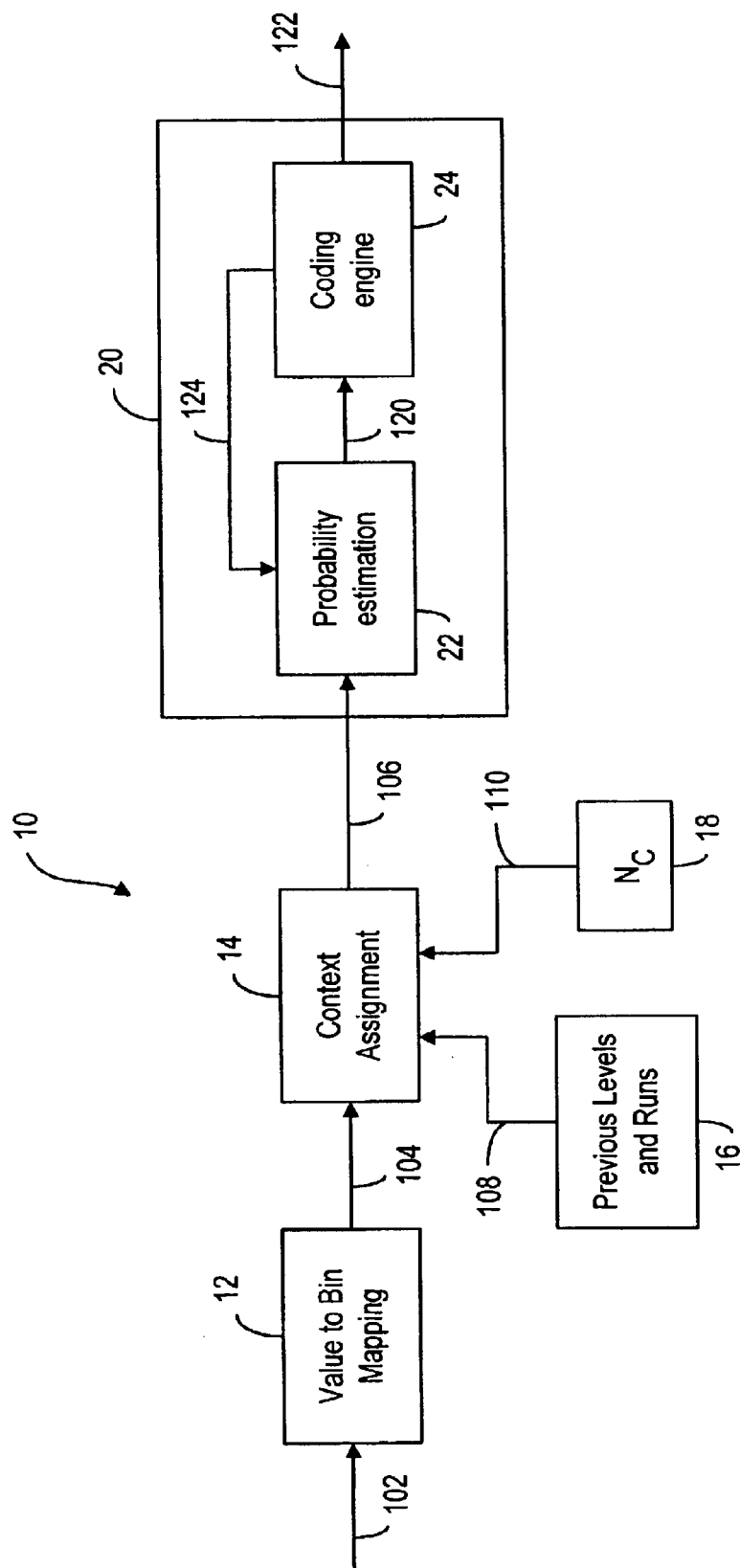


FIG. 9

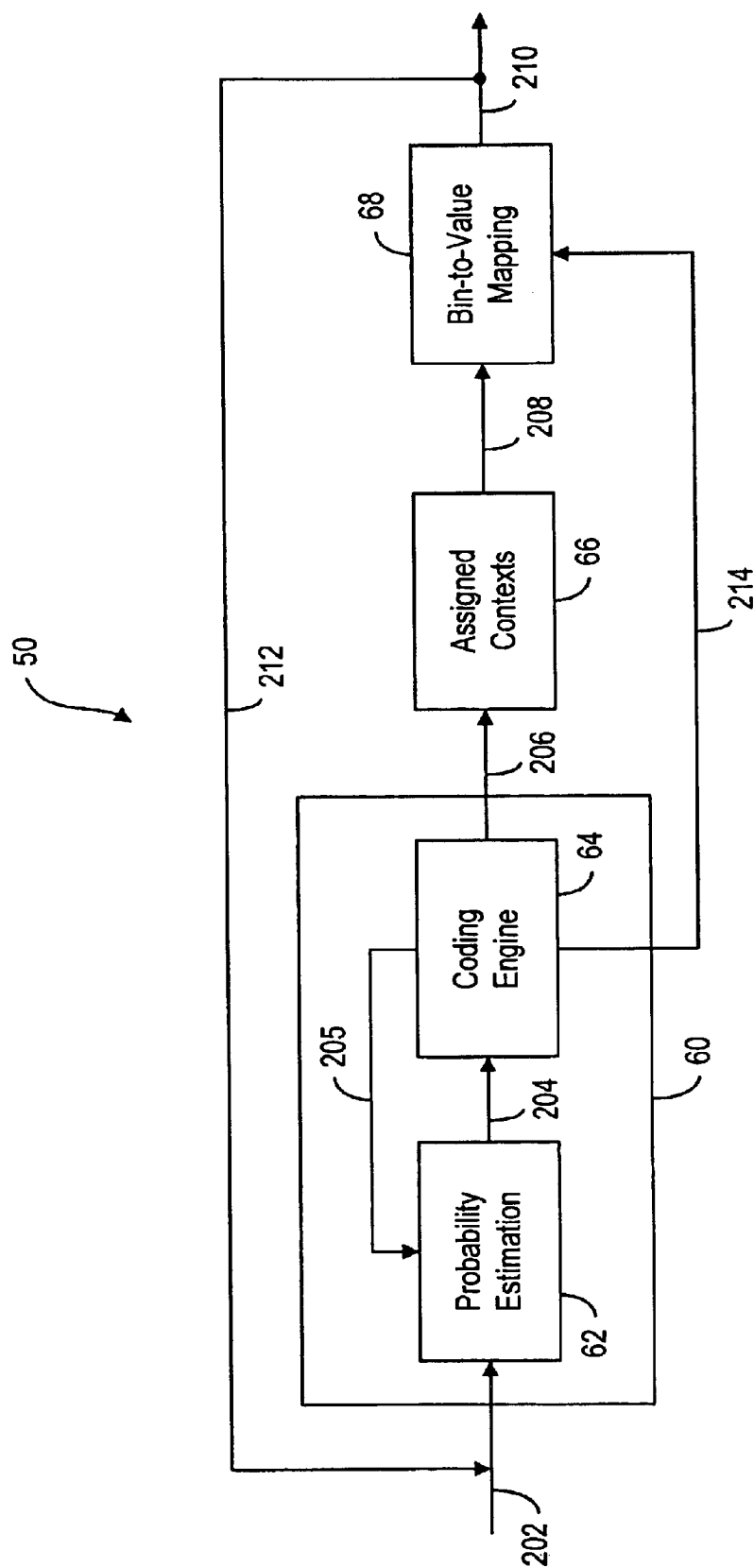
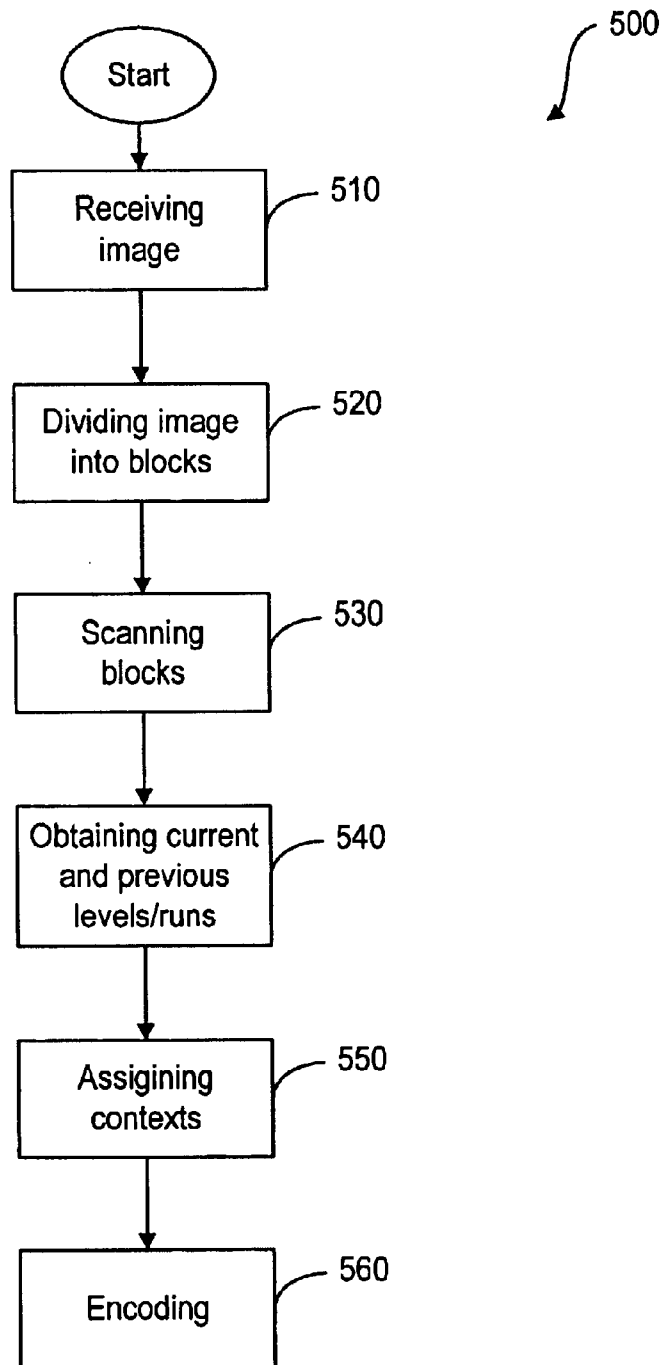


FIG. 10

**FIG. 11**

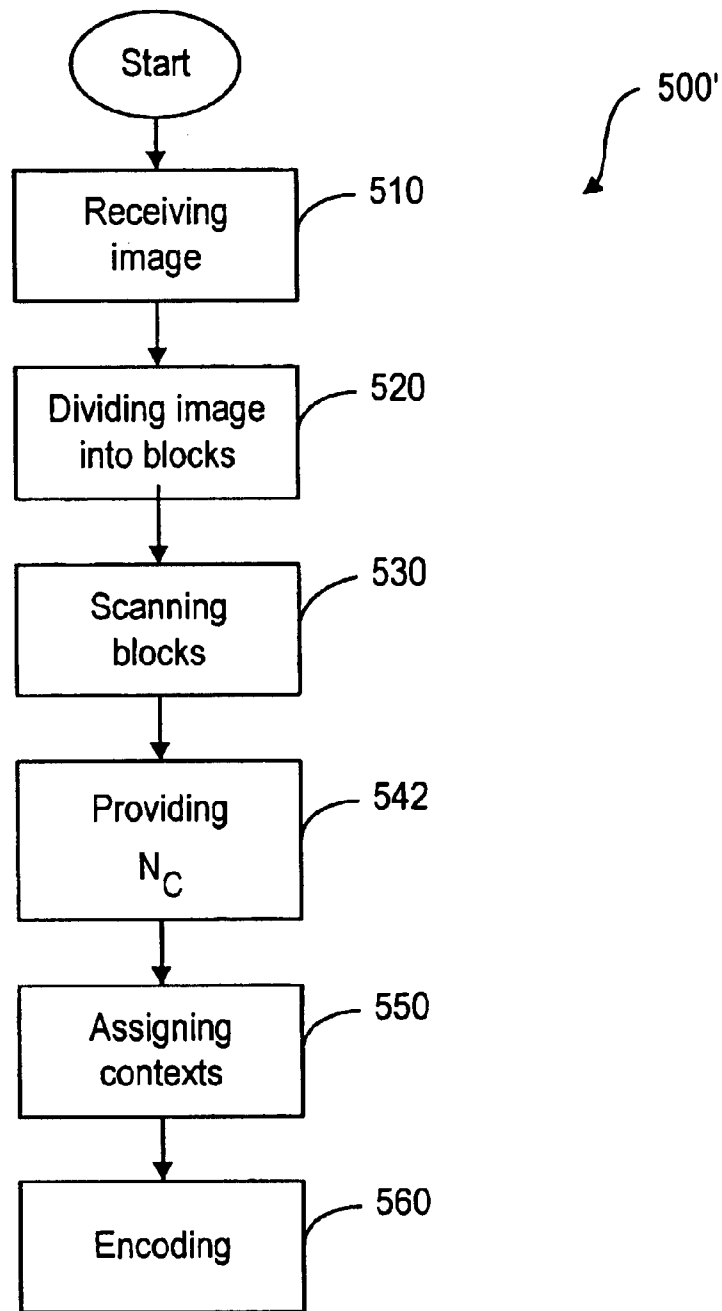


FIG. 12

1

METHOD AND SYSTEM FOR CONTEXT-BASED ADAPTIVE BINARY ARITHMETIC CODING

This patent application is based on and claims priority to U.S. Provisional Application No. 60/322,112, filed Sep. 14, 2001.

FIELD OF THE INVENTION

The present invention relates generally to the compression of still images and video sequences and, more particularly, to a method and system for context-based adaptive binary arithmetic coding.

BACKGROUND OF THE INVENTION

A digital image in uncompressed form comprises an array of image pixels or picture elements. For example, in a commonly used digital image format, known as the Quarter Common Interchange Format (QCIF), an image, or frame, comprises 25,344 pixels arranged in an array 176×144 pixels. Each pixel, in turn, is represented by a certain number of bits, which carry information about the brightness (luminance) and/or color (chrominance) of the pixel. Different schemes exist for representing the luminance and/or chrominance of pixels in a digital image. Commonly, a so-called YUV color model is used. The luminance, or Y, component represents the luminance of the pixel, while the color of the pixel is represented by two chrominance or color difference components, labelled U and V. Other color models, such as RGB (Red, Green, Blue) color models, which are based on components representing the three primary colors of light, are also commonly used. However, color models based on a luminance/chrominance representation provide advantages compared with color models based on primary colors. These stem from the nature of the human visual system, which is more sensitive to intensity variations than it is to color variations. YUV color models typically exploit this property by using a lower spatial resolution for the chrominance components (U, V) than for the luminance component (Y). In this way the amount of information needed to represent the color information in an image can be reduced without a noticeable reduction in perceived image quality.

The lower spatial resolution of the chrominance components is usually attained by sub-sampling. Typically, a block of 16×16 image pixels is represented by four blocks of 8×8 pixels comprising luminance information and the corresponding chrominance components are each represented by one block of 8×8 pixels representing an area of the image equivalent to that of the 16×16 pixels in the luminance component. The chrominance components are thus spatially sub-sampled by a factor of 2 in the x and y directions. The resulting assembly of four 8×8 pixel luminance blocks and two spatially corresponding 8×8 pixel chrominance blocks is commonly referred to as a YUV macroblock, or macroblock, for short. A QCIF image comprises 11×9 such macroblocks. If the luminance blocks and chrominance blocks are represented with 8 bit resolution (that is by numbers in the range 0 to 255), the total number of bits required to represent the luminance and chrominance information associated with each macroblock is $6 \times (8 \times 8 \times 8) = 3072$ bits. Thus, the number of bits needed to represent an image in QCIF format is $99 \times 3072 = 304,128$ bits.

It should be appreciated that even in the situation described above, where both chrominance components of a digital color image are sub-sampled by a factor of two, an

2

uncompressed image of only moderate size (e.g. 176×144 pixels) requires a large number of bits for its representation. This means that the amount of memory required to store digital images in uncompressed form is excessive. Furthermore, if still images are to be transferred, for example over a data communications network having a moderate or low available bandwidth, transmission times may become lengthy, or the network may become congested. Bandwidth requirements are even more severe if it is desired to transmit a series of images as a digital video sequence in real time. For example, transmission of a digital video sequence comprising a series of images in uncompressed QCIF format, represented using a YUV color model, at a rate of 30 frames per second, requires more than 9 Mbits/s (million bits per second). Such a high data rate is generally impractical for use in video recording, transmission and display applications because of the very large storage capacity, transmission channel capacity and hardware performance required. If a video sequence is to be transmitted in real-time over a fixed line network such as an ISDN (Integrated Services Digital Network) or a PSTN (Public Service Telephone Network), the available data transmission bandwidth is typically of the order of 64 kbits/s. In mobile video-telephony, where transmission takes place at least in part over a radio communications link, the available bandwidth can be as low as 20 kbits/s. This means that a significant reduction in the amount of information used to represent video data must be achieved in order to enable transmission of digital images or video sequences over low bandwidth communication networks. It is nevertheless desirable that this reduction should be achieved without significantly degrading the quality of the images/video sequence.

Over the past years, a considerable amount of research work has been directed towards reducing the amount of data required to represent digital images and video sequences, resulting in the development of numerous different schemes and international standards for compressing digital still images and digital video. The basic approach to image compression used in almost all still image and video encoders existing today involves block-based transform coding. Typically, transform coding translates image data from a representation comprising pixel values to a form comprising a set of coefficient values, each of which is a weighting factor (multiplier) for a basis function of the transform in question. It can be shown that there is a considerable degree of spatial redundancy within a typical digital image. In practical terms, this means that in general the value of any pixel within an image is substantially the same as the value of other pixels in its immediate vicinity; that is, there is a significant degree of correlation between pixel values. It is further known that when certain mathematical transformations, such as the two-dimensional Discrete Cosine Transform (DCT), are performed on image data, this spatial redundancy is reduced significantly, thereby producing a more compact representation of the image data. Block-based Transform Coding as Used in JPEG Still Image Coding

In still image compression, such as that performed according to the baseline mode of the widely-used JPEG standard, an image to be coded is first divided into an array of non-overlapping square blocks, each block comprising, for example, an 8×8 array of image pixels. In the case of the JPEG baseline, a two-dimensional Discrete Cosine Transform (DCT) is then applied independently to each of the image blocks. This has the effect of converting the image data from the pixel value domain to the spatial frequency

domain and to produce a corresponding set of coefficient values, each of which is a weighting factor for a basis function of the two-dimensional DCT. The coefficient values thus produced are quantized and then coded in a lossless manner using entropy coding to further reduce the amount of data (i.e. number of bits) required for their representation. According to the JPEG baseline, the entropy coder employs only Huffman coding to produce a compressed bit-stream, although in other modes arithmetic coding may alternatively be used. Finally, data describing image and coding parameters (e.g. type of compression, quantization and coding tables, image size, etc.) is embedded in the bit-stream produced by the entropy encoder. As the JPEG standard comprises four alternative coding modes and places few constraints on the quantization and coding tables that can be used, this is necessary in order to enable JPEG compressed bit-streams to be interchanged among different platforms and for images to be reconstructed without any ambiguity.

A digital video sequence, like an ordinary motion picture recorded on film, comprises a sequence of still images (often referred to as 'frames'), the illusion of motion being created by displaying the frames one after the other at a relatively fast rate, typically 15 to 30 frames per second. As in any still image, the pixel values of an individual frame within a digital video sequence exhibit considerable spatial redundancy. Therefore, the frames of a digital video sequence are amenable to block-based transform coding, just like individual still images.

Images in the consecutive frames of a video sequence also tend to be quite similar and thus the overall change between one video frame and the next is rather small. This means that there is considerable temporal redundancy within a typical digital video sequence. For example, a scene may comprise some stationary elements, such as background scenery, and some moving areas, for example the face of a newsreader. In consecutive frames of the sequence, it is likely that the background will remain unaltered and the only movement in the scene will be due to changes in facial expression of the newsreader. Thus, when forming a compressed representation of a video sequence there is also a possibility to use techniques which reduce the temporal redundancy of the image data of the sequence in addition to methods that reduce spatial redundancy, thereby allowing further data compression to be achieved.

Hybrid Video Encoder/Decoder

State of the art video coding systems make use of a technique known as 'motion-compensated prediction', to reduce the temporal redundancy in video sequences. Using motion-compensated prediction, the image content of some (often many) frames in a digital video sequence is 'predicted' from one or more other frames in the sequence, known as 'reference frames'. Prediction of image content is achieved by tracing the motion of objects or regions of an image between a frame to be coded (compressed) and the reference frame(s) using 'motion vectors'. In general, the reference frame(s) may precede the frame to be coded or may follow it in the video sequence. However, as will become apparent from discussions later in the text, it is not appropriate (or possible) to apply motion-compensated prediction to all frames of a video sequence and thus at least two types of encoding are used in state of the art video coding systems.

Frames of a video sequence which are compressed using motion-compensated prediction are generally referred to as INTER-coded or P-frames. Motion-compensated prediction alone rarely provides a sufficiently precise representation of the image content of a video frame and therefore it is

typically necessary to provide a so-called 'prediction error' (PE) frame with each INTER-coded frame. As will be explained in greater detail later in the text, the prediction error frame represents the difference between a decoded version of the INTER-coded frame and the image content of the frame to be coded. More specifically, the prediction error frame comprises values that represent the difference between pixel values in the frame to be coded and corresponding reconstructed pixel values formed on the basis of a predicted (INTER-coded) version of the frame in question. Consequently, the prediction error frame has characteristics similar to a still image and block-based transform coding can be applied in order to reduce the amount of data (number of bits) required to represent it.

Frames of a video sequence which are not compressed using motion-compensated prediction are referred to as INTRA-coded or I-frames. Generally, INTRA-coded frames are produced by applying block-based transform coding directly to the pixel values of the frame to be coded. Additionally, where possible, blocks of INTRA-coded frames are predicted from previously coded blocks within the same frame. This technique, known as INTRA-prediction, has the effect of further reducing the amount of data required to represent an INTRA-coded frame.

In order to illustrate principles of block-based transform coding and motion-compensated prediction in greater detail, reference will now be made to FIG. 1, which is a schematic of a generic hybrid video encoder that employs a combination of INTRA- and INTER-coding to produce a compressed (encoded) video bit-stream. A corresponding decoder is illustrated in FIG. 2 and will be described later in the text.

The video encoder 300 comprises an input 301 for receiving a digital video signal from a camera or other video source (not shown). It also comprises a transformation unit 304 which is arranged to perform a block-based discrete cosine transform (DCT), a quantizer 306, an inverse quantizer 308, an inverse transformation unit 310, arranged to perform an inverse block-based discrete cosine transform (IDCT), combiners 312 and 316, and a frame store 320. The encoder further comprises a motion estimator 330, a motion field coder 340 and a motion compensated predictor 350. Switches 302 and 314 are operated co-operatively by control manager 360 to switch the encoder between an INTRA-mode of video encoding and an INTER-mode of video encoding. The encoder 300 also comprises a video multiplex coder 370 which forms a single bit-stream from the various types of information produced by the encoder 300 for further transmission to a remote receiving terminal or, for example, for storage on a mass storage medium, such as a computer hard drive (not shown).

Encoder 300 operates as follows. Each frame of uncompressed video provided from the video source to input 301 is received and processed macroblock-by-macroblock, preferably in raster-scan order. When the encoding of a new video sequence starts, the first frame of the sequence is encoded as an INTRA-coded frame. Subsequently, the encoder is programmed to code each frame in INTER-coded format, unless one of the following conditions is met: 1) it is judged that the current frame being coded is so dissimilar from the reference frame used in its prediction that excessive prediction error information is produced; 2) a predefined INTRA frame repetition interval has expired; or 3) feedback is received from a receiving terminal indicating a request for a frame to be provided in INTRA-coded format.

The occurrence of condition 1) is detected by monitoring the output of the combiner 316. The combiner 316 forms a difference between the current macroblock of the frame

5

being coded and its prediction, produced in the motion compensated prediction block **350**. If a measure of this difference (for example a sum of absolute differences of pixel values) exceeds a predetermined threshold, the combiner **316** informs the control manager **360** via a control line **319** and the control manager **360** operates the switches **302** and **314** via control line **313** so as to switch the encoder **300** into INTRA-coding mode. Occurrence of condition 2) is monitored by means of a timer or frame counter implemented in the control manager **360**, in such a way that if the timer expires, or the frame counter reaches a predetermined number of frames, the control manager **360** operates the switches **302** and **314** via control line **313** to switch the encoder into INTRA-coding mode. Condition 3) is triggered if the control manager **360** receives a feedback signal from, for example, a receiving terminal, via control line **321** indicating that an INTRA frame refresh is required by the receiving terminal. Such a condition may arise, for example, if a previously transmitted frame is badly corrupted by interference during its transmission, rendering it impossible to decode at the receiver. In this situation, the receiving decoder issues a request for the next frame to be encoded in INTRA-coded format, thus re-initialising the coding sequence.

Operation of the encoder **300** in INTRA-coding mode will now be described. In INTRA-coding mode, the control manager **360** operates the switch **302** to accept video input from input line **318**. The video signal input is received macroblock by macroblock from input **301** via the input line **318**. As they are received, the blocks of luminance and chrominance values which make up the macroblock are passed to the DCT transformation block **304**, which performs a 2-dimensional discrete cosine transform on each block of values, producing a 2-dimensional array of DCT coefficients for each block. In a situation such as that described earlier, where each macroblock comprises four 8x8 pixel blocks of luminance values and two spatially corresponding 8x8 pixel blocks of chrominance values, DCT transformation block **304** produces an 8x8 array of coefficient values for each block.

The DCT coefficients for each block are passed to the quantizer **306**, where they are quantized using a quantization parameter QP. Selection of the quantization parameter QP is controlled by the control manager **360** via control line **315**. Quantization introduces a loss of information, as the quantized coefficients have a lower numerical precision than the coefficients originally generated by the DCT transformation block **304**. This provides a further mechanism by which the amount of data required to represent each image of the video sequence can be reduced. However, unlike the DCT transformation, which is essentially lossless, the loss of information introduced by quantization causes an irreversible degradation in image quality. The greater the degree of quantization applied to the DCT coefficients, the greater the loss of image quality.

The quantized DCT coefficients for each block are passed from the quantizer **306** to the video multiplex coder **370**, as indicated by line **325** in FIG. 1. The video multiplex coder **370** orders the transform coefficients for each block using a zigzag scanning procedure. This operation converts the two-dimensional array of quantized transform coefficients into a one-dimensional array. Typical zigzag scanning orders, such as that shown in FIG. 3, order the coefficients approximately in ascending order of spatial frequency. This also tends to order the coefficients according to their values, such that coefficients positioned earlier in the one-dimensional array are more likely to have larger absolute

6

values than coefficients positioned later in the array. This is because lower spatial frequencies tend to have higher amplitudes within the image blocks. Consequently, the last values in the one-dimensional array of quantized transform coefficients are commonly zeros.

Run-Level Coding of DCT Transform Coefficients

Typically, the video multiplex coder **370** represents each non-zero quantized coefficient in the one dimensional array by two values, referred to as level and run. Level is the value of the quantized coefficient and run is the number of consecutive zero-valued coefficients preceding the coefficient in question. The run and level values for a given coefficient are ordered such that the level value precedes the associated run value. A level value equal to zero is used to indicate that there are no more non-zero coefficient values in the block. This 0-level value is referred to as an EOB (end-of-block) symbol.

Entropy Coding

The run and level values are further compressed in the video multiplex coder **370** using entropy coding. Entropy coding is a lossless operation, which exploits the fact that symbols within a data set to be coded generally have different probabilities of occurrence. Therefore, instead of using a fixed number of bits to represent each symbol, a variable number of bits is assigned such that symbols which are more likely to occur are represented by code-words having fewer bits. For this reason, entropy coding is often referred to as Variable Length Coding (VLC). Since certain values of levels and runs are more likely than other values to occur, entropy coding techniques can be used effectively to reduce the number of bits required to represent the run and level values. A number of different methods can be used to implement entropy coding. For example, entropy coding of the run and level parameters may be implemented by means of look-up tables which define the mapping between each possible symbol in the data set to be coded and its corresponding variable length code. Such look-up tables are often defined by statistical analysis of training material comprising symbols identical to those to be coded and having similar statistical properties. An alternative technique, known as arithmetic coding, can also be used to convert the run and level values into variable length code-words. In arithmetic coding a group of symbols, for example the run and level values for a block of quantized transform coefficients, are coded as a floating point decimal number.

Once the run and level values have been entropy coded using an appropriate method, the video multiplex coder further combines them with control information, also entropy coded using a variable length coding method appropriate for the kind of information in question, to form a single compressed bit-stream of coded image information **335**.

A locally decoded version of the macroblock is also formed in the encoder **300**. This is done by passing the quantized transform coefficients for each block, output by quantizer **306**, through inverse quantizer **308** and applying an inverse DCT transform in inverse transformation block **310**. In this way a reconstructed array of pixel values is constructed for each block of the macroblock. The resulting decoded image data is input to combiner **312**. In INTRA-coding mode, switch **314** is set so that the input to the combiner **312** via switch **314** is zero. In this way, the operation performed by combiner **312** is equivalent to passing the decoded image data unaltered.

As subsequent macroblocks of the current frame are received and undergo the previously described encoding and decoding steps in blocks **304**, **306**, **308**, **310** and **312**, a

7

decoded version of the INTRA-coded frame is built up in frame store **320**. When the last macroblock of the current frame has been INTRA-coded and subsequently decoded, the frame store **320** contains a completely decoded frame, available for use as a prediction reference frame in coding a

subsequently received video frame in INTER-coded format. Operation of the encoder **300** in INTER-coding mode will now be described. In INTER-coding mode, the control manager **360** operates switch **302** to receive its input from line **317**, which comprises the output of combiner **316**. The combiner **316** receives the video input signal macroblock by macroblock from input **301**. As combiner **316** receives the blocks of luminance and chrominance values which make up the macroblock, it forms corresponding blocks of prediction error information. The prediction error information represents the difference between the block in question and its prediction, produced in the motion compensated prediction block **350**. More specifically, the prediction error information for each block of the macroblock comprises a two-dimensional array of values, each of which represents the difference between a pixel value in the block of luminance or chrominance information being coded and a decoded pixel value obtained by forming a motion-compensated prediction for the block, according to the procedure described below. Thus, in a situation where each macroblock comprises four 8×8 pixel blocks of luminance values and two spatially corresponding 8×8 pixel blocks of chrominance values, the prediction error information for the macroblock similarly comprises four 8×8 blocks of luminance prediction error values and two spatially corresponding 8×8 blocks of chrominance prediction error values.

The prediction error information for each block of the macroblock is passed to DCT transformation block **304**, which performs a two-dimensional discrete cosine transform on each block of prediction error values to produce a two-dimensional array of DCT transform coefficients for each block. Thus, in a situation where the prediction error information for each macroblock comprises four 8×8 blocks of luminance prediction error values and two spatially corresponding 8×8 blocks of chrominance prediction error values, DCT transformation block **304** produces an 8×8 array of transform coefficient values for each prediction error block. The transform coefficients for each prediction error block are passed to quantizer **306** where they are quantized using a quantization parameter QP, in a manner analogous to that described above in connection with operation of the encoder in INTRA-coding mode. Again, selection of the quantization parameter QP is controlled by the control manager **360** via control line **315**.

The quantized DCT coefficients representing the prediction error information for each block of the macroblock are passed from quantizer **306** to video multiplex coder **370**, as indicated by line **325** in FIG. 1. As in INTRA-coding mode, the video multiplex coder **370** orders the transform coefficients for each prediction error block using the previously described zigzag scanning procedure (see FIG. 3) and then represents each non-zero quantized coefficient as a level and a run value. It further compresses the run and level values using entropy coding, in a manner analogous to that described above in connection with INTRA-coding mode. Video multiplex coder **370** also receives motion vector information (described in the following) from motion field coding block **340** via line **326** and control information from control manager **360**. It entropy codes the motion vector information and forms a single bit-stream of coded image information, **335** comprising the entropy coded motion vector, prediction error and control information.

8

The quantized DCT coefficients representing the prediction error information for each block of the macroblock are also passed from quantizer **306** to inverse quantizer **308**. Here they are inverse quantized and the resulting blocks of inverse quantized DCT coefficients are applied to inverse DCT transform block **310**, where they undergo inverse DCT transformation to produce locally decoded blocks of prediction error values. The locally decoded blocks of prediction error values are then input to combiner **312**. In INTER-coding mode, switch **314** is set so that the combiner **312** also receives predicted pixel values for each block of the macroblock, generated by motion-compensated prediction block **350**. The combiner **312** combines each of the locally decoded blocks of prediction error values with a corresponding block of predicted pixel values to produce reconstructed image blocks and stores them in frame store **320**.

As subsequent macroblocks of the video signal are received from the video source and undergo the previously described encoding and decoding steps in blocks **304**, **306**, **308**, **310**, **312**, a decoded version of the INTER-coded frame is built up in frame store **320**. When the last macroblock of the frame has been INTER-coded and subsequently decoded, the frame store **320** contains a completely decoded frame, available for use as a prediction reference frame in encoding a subsequently received video frame in INTER-coded format.

Formation of a prediction for a macroblock of the current frame will now be described. Any frame encoded in INTER-coded format requires a reference frame for motion-compensated prediction. This means, necessarily, that when encoding a video sequence, the first frame to be encoded, whether it is the first frame in the sequence, or some other frame, must be encoded in INTRA-coded format. This, in turn, means that when the video encoder **300** is switched into INTER-coding mode by control manager **360**, a complete reference frame, formed by locally decoding a previously encoded frame, is already available in the frame store **320** of the encoder. In general, the reference frame is formed by locally decoding either an INTRA-coded frame or an INTER-coded frame.

The first step in forming a prediction for a macroblock of the current frame is performed by motion estimation block **330**. The motion estimation block **330** receives the blocks of luminance and chrominance values which make up the current macroblock of the frame to be coded via line **328**. It then performs a block matching operation in order to identify a region in the reference frame which corresponds substantially with the current macroblock. In order to perform the block matching operation, motion field estimation block accesses reference frame data stored in frame store **320** via line **327**. More specifically, motion estimation block **330** performs block-matching by calculating difference values (e.g. sums of absolute differences) representing the difference in pixel values between the macroblock under examination and candidate best-matching regions of pixels from a reference frame stored in the frame store **320**. A difference value is produced for candidate regions at all possible offsets within a predefined search region of the reference frame and motion estimation block **330** determines the smallest calculated difference value. The offset between the macroblock in the current frame and the candidate block of pixel values in the reference frame that yields the smallest difference value defines the motion vector for the macroblock in question.

Once the motion estimation block **330** has produced a motion vector for the macroblock, it outputs the motion vector to the motion field coding block **340**. The motion field

coding block **340** approximates the motion vector received from motion estimation block **330** using a motion model comprising a set of basis functions and motion coefficients. More specifically, the motion field coding block **340** represents the motion vector as a set of motion coefficient values which, when multiplied by the basis functions, form an approximation of the motion vector. Typically, a translational motion model having only two motion coefficients and basis functions is used.

The motion coefficients are passed from motion field coding block **340** to motion compensated prediction block **350**. Motion compensated prediction block **350** also receives the best-matching candidate region of pixel values identified by motion estimation block **330** from frame store **320**. Using the approximate representation of the motion vector generated by motion field coding block **340** and the pixel values of the best-matching candidate region of pixels from the reference frame, motion compensated prediction block **350** generates an array of predicted pixel values for each block of the macroblock. Each block of predicted pixel values is passed to combiner **316** where the predicted pixel values are subtracted from the actual (input) pixel values in the corresponding block of the current macroblock. In this way a set of prediction error blocks for the macroblock is obtained.

Operation of the video decoder **400**, shown in FIG. 2 will now be described. The decoder **400** comprises a video multiplex decoder **470**, which receives an encoded video bit-stream **335** from the encoder **300** and demultiplexes it into its constituent parts, an inverse quantizer **410**, an inverse DCT transformer **420**, a motion compensated prediction block **440**, a frame store **450**, a combiner **430**, a control manager **460**, and an output **480**.

The control manager **460** controls the operation of the decoder **400** in response to whether an INTRA- or an INTER-coded frame is being decoded. An INTRA/INTER trigger control signal, which causes the decoder to switch between decoding modes is derived, for example, from picture type information provided in a header portion of each compressed video frame received from the encoder. The INTRA/INTER trigger control signal is extracted from the encoded video bit-stream by the video multiplex decoder **470** and is passed to control manager **460** via control line.

Decoding of an INTRA-coded frame is performed on a macroblock-by-macroblock basis, each macroblock being decoded substantially as soon as encoded information relating to it is identified in the received video bit-stream **335**. The video multiplex decoder **470** first separates the encoded information for the blocks of the macroblock from possible control information relating to the macroblock in question. The encoded information for each block of an INTRA-coded macroblock comprises variable length code-words. These code-words represent the entropy coded level and run values for the non-zero quantized DCT coefficients of the block. The video multiplex decoder **470** decodes the variable length code-words using a variable length decoding method corresponding to the encoding method used in the encoder **300** and thereby recovers the level and run values. It then reconstructs the array of quantized transform coefficient values for each block of the macroblock and passes them to inverse quantizer **410**. Any control information relating to the macroblock is also decoded in the video multiplex decoder using an appropriate variable length decoding method and is passed to control manager **460**. In particular, information relating to the level of quantization applied to the transform coefficients is extracted from the encoded bit-stream by video multiplex decoder **470** and is provided to control manager **460** via control line **324**. The control

manager, in turn, conveys this information to inverse quantizer **420** via control line. Inverse quantizer **410** inverse quantizes the quantized DCT coefficients for each block of the macroblock according to the control information and provides the now inverse quantized DCT coefficients to inverse DCT transformer **420**.

Inverse DCT transformer **420** performs an inverse DCT transform on the inverse quantized DCT coefficients for each block of the macroblock to form a decoded block of image information comprising reconstructed pixel values. As motion-compensated prediction is not used in the encoding/decoding of INTRA-coded macroblocks, control manager **460** controls combiner **430** in such a way as to prevent any reference information being used in the decoding of the INTRA-coded macroblock. The reconstructed pixel values for each block of the macroblock are passed to the video output **480** of the decoder where, for example, they can be provided to a display device (not shown). The reconstructed pixel values for each block of the macroblock are also stored in frame store **450**. As subsequent macroblocks of the INTRA-coded frame are decoded and stored, a decoded frame is progressively assembled in the frame store **450** and thus becomes available for use as a reference frame for motion compensated prediction in connection with the decoding of subsequently received INTER-coded frames.

INTER-coded frames are also decoded macroblock by macroblock, each INTER-coded macroblock being decoded substantially as soon as encoded information relating to it is identified in the received bit-stream. The video multiplex decoder **470** separates the encoded prediction error information for each block of the INTER-coded macroblock from encoded motion vector information and possible control information relating to the macroblock in question. As explained in the foregoing, the encoded prediction error information for each block of the macroblock comprises variable length codewords which represent the entropy coded level and run values for the non-zero quantized transform coefficients for the prediction error block in question. The video multiplex decoder **470** decodes the variable length code-words using a variable length decoding method corresponding to the encoding method used in the encoder **300** and thereby recovers the level and run values. It then reconstructs an array of quantized transform coefficient values for each prediction error block and passes them to inverse quantizer **410**. Control information relating to the INTER-coded macroblock is also decoded in the video multiplex decoder using an appropriate variable length decoding method and is passed to control manager **460**. Information relating to the level of quantization applied to the transform coefficients of the prediction error blocks is extracted from the encoded bit-stream and provided to control manager **460** via control line **324**. The control manager, in turn, conveys this information to inverse quantizer **410** via control line. Inverse quantizer **410** inverse quantizes the quantized DCT coefficients representing the prediction error information for each block of the macroblock according to the control information and provides the now inverse quantized DCT coefficients to inverse DCT transformer **420**. The inverse quantized DCT coefficients representing the prediction error information for each block are then inverse transformed in the inverse DCT transformer **420** to yield an array of reconstructed prediction error values for each block of the macroblock.

The encoded motion vector information associated with the macroblock is extracted from the encoded video bit-stream **335** by video multiplex decoder **470** and is decoded using an appropriate variable length decoding method. The

decoded motion vector information thus obtained is passed via data line 326 to motion compensated prediction block 440, which reconstructs a motion vector for the macroblock using the same motion model as that used to encode the INTER-coded macroblock in encoder 300. The reconstructed motion vector approximates the motion vector originally determined by motion estimation block 330 of the encoder. The motion compensated prediction block 440 of the decoder uses the reconstructed motion vector to identify the location of a region of reconstructed pixels in a prediction reference frame stored in frame store 450. The reference frame may be, for example, a previously decoded INTRA-coded frame, or a previously decoded INTER-coded frame. In either case, the region of pixels indicated by the reconstructed motion vector is used to form a prediction for the macroblock in question. More specifically, the motion compensated prediction block 440 forms an array of pixel values for each block of the macroblock by copying corresponding pixel values from the region of pixels identified in the reference frame. The prediction, that is the blocks of pixel values derived from the reference frame, are passed from motion compensated prediction block 440 to combiner 430 where they are combined with the decoded prediction error information. In practice, the pixel values of each predicted block are added to corresponding reconstructed prediction error values output by inverse DCT transformer 420. In this way an array of reconstructed pixel values for each block of the macroblock is obtained. The reconstructed pixel values are passed to the video output 480 of the decoder and are also stored in frame store 450. As subsequent macroblocks of the INTER-coded frame are decoded and stored, a decoded frame is progressively assembled in the frame store 450 and thus becomes available for use as a reference frame for motion-compensated prediction of other INTER-coded frames.

H.26L Video Coding Standard

ITU-T recommendation H.26L is the latest in a family of video coding standards developed by the International Telecommunications Union. It is intended in particular for video coding at very low bit rates, typically below 64 kbits/s, which makes it especially suitable for the coding of digital video for transmission via radio communication networks or any fixed line communication network in which optimal use of available bandwidth is a priority. The video encoding system defined by ITU-T H.26L is a hybrid video coding system, which operates according to the general principles described above in connection with the generic video encoder 300 and decoder 400 illustrated in FIGS. 1 and 2. In particular, a video encoding system implemented according to H.26L employs a combination of block-based transform coding and motion-compensated prediction to reduce the spatial and temporal redundancy within video sequences.

The latest version of the H.26L recommendation, known as Test Model 8 (TML8) and described in "H.26L Test Model Long Term Number 8 (TML-8) draft0" (ITU-T Telecommunications Standardization Section, Study Group 16, Video Coding Experts Group), specifies two alternative entropy coding modes. In the first (default) mode a so-called Universal Variable Length Coding (UVLC) method is used to encode all syntax elements. The UVLC coding mode is a look-up table method in which the same set of variable length code-words is used to represent all the different kinds of information produced by the video encoder, regardless of the type of information in question. The alternative entropy coding method, specified for use in the so-called high complexity profile of H.26L, is a technique known as Context-based Adaptive Binary Arithmetic Coding (CABAC). This is a form of binary arithmetic coding which

continually adapts to the statistical properties of the information being coded and is known in the art to be one of the most efficient forms of entropy coding (see H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, pp. 520-540, June 1987).

Because UVLC entropy coding uses the same set of variable length code-words to represent all types of information produced by the video encoder, in general the statistical properties of the code-words do not match optimally with the characteristics of the information to be encoded. For example, the frequency of occurrence of particular run and level values used to represent the quantized DCT coefficients for an INTRA-coded image block is likely to be different from the occurrence of values in control information relating to quantization parameter values. The CABAC entropy coding method was introduced into the H.26L recommendation in order to overcome the inherently sub-optimal nature of the UVLC entropy coding method. As explained earlier in the text, arithmetic coding represents a group of symbols to be coded with a single variable length code (a floating-point number). This provides particular advantages compared with entropy coding methods, which encode each symbol independently. Specifically, entropy coding methods which encode each symbol independently require at least one bit to represent each symbol. Because arithmetic coding represents groups of symbols with a single code-word, it is possible to achieve data compression rates of less than one bit per symbol. Thus the CABAC method provided in H.26L also provides the possibility of improved data compression. Furthermore, because it is an adaptive method, it is also able to take into account changes in the statistical characteristics of the information being coded, ensuring that data compression performance is maintained even if the nature of the data being encoded changes to some extent.

Context-Based Arithmetic Coding

As explained above, CABAC arithmetic coding is an entropy coding method, which is able to adapt to changing statistics of the information to be encoded. In this way it is capable of providing improved compression efficiency compared with entropy coding techniques which assume fixed statistical properties. FIG. 4 illustrates an exemplary context-based binary arithmetic encoder 700. CABAC is a binary arithmetic coding method and thus data symbols to be coded which have non-binary values are first converted to binary values ('binarized') in binary mapping block 710. The binarization process involves mapping a symbol to be coded to a sequence of bins, each of which has a corresponding bin number and can be assigned a value of either 0 or 1. An example of such a mapping is given below in Table 1. In principle other binarization schemes can be envisaged.

TABLE 1

value	Bin Sequence							
0	1							
1	0	1						
2	0	0	1					
3	0	0	0	1				
4	0	0	0	0	1			
5	0	0	0	0	0	1		
6	0	0	0	0	0	0	1	
...
bin_nr.	1	2	3	4	5	6	7	.

In the CABAC method each of the bins is assigned to a so-called 'context' (hence the name context-based arithmetic coding). A context can be thought of as grouping together

bins, which have similar statistical characteristics. In other words, each bin assigned to a particular context is assumed to have a similar probability of containing the value 1 or 0 as the other bins belonging to that context. In this way, the probability estimates used to generate code-words in the arithmetic coder are defined for each context rather than for each possible bin to be encoded. Each context is defined according to a 'context model', established in advance and based on information about the statistical characteristics of the data symbols (and thus the bins) to be encoded. Generally, the data compression ratio achieved by a binary arithmetic encoder is enhanced if the difference between the probability of occurrence of a 0 and probability of occurrence of a 1 is maximised. In a similar way, the performance of a context-based arithmetic coding also depends on the choice of context model. This means that, in general, context models should be chosen so as to maximise the difference between the probability of occurrence of 0's and 1's for the bins assigned to each context.

In the exemplary context-based arithmetic coder illustrated in FIG. 4, once a symbol to be coded has been binarized in binary mapping block 710 it is assigned to a corresponding context in context assignment block 720. The value assigned to the corresponding bin (i.e., 1 or 0) is then passed to the arithmetic coder 730. The coding engine 750 of arithmetic coder 730 then encodes the bin value using a probability estimate for the context to which the bin is assigned. The performance, that is the data compression ratio achieved by the arithmetic encoder, depends on the accuracy of the probability estimates. In principle, the estimates may be fixed or adaptive. If fixed probability estimates are used, the probability estimates for each context are assigned to predefined values and remain unchanged during the encoding process. Fixed probability estimates are typically obtained in advance by analysing training material having statistical properties similar to those of the actual data to be encoded. If adaptive probability estimates are used, fixed values are used to initialise the probability estimates for each context and the probabilities are then updated throughout the encoding process based on the actual statistical properties of the data (bins) encoded so far. Adaptive probability estimates generally perform better since they can adjust to the material being encoded.

The exemplary context-based arithmetic coder illustrated in FIG. 4 employs adaptive probability estimates and comprises a probability estimation block 740 where updated probability estimates are calculated. The probability estimates for each context are updated by keeping a record of the number of occurrences of 1 and 0 for each of the bins assigned to each context. For example, if the bins assigned to an arbitrary context k have been assigned the value 0 m times and the value 1 n times, then the probability estimate for 1 in context k is $n/(n(m+1))$ and the probability estimate for 0 is $(m+1)/(n(m+1))$.

FIG. 5 illustrates a context-based arithmetic decoder 800 corresponding to the encoder described in connection with FIG. 4. A bit-stream representing arithmetic coded data symbols is received by the context-based arithmetic decoder at input 810. Initially, based on the previously decoded symbols, a context is calculated in a context assignment block 850 and the probability estimates of the bin values are updated. Context assignment, as carried out in the context assignment block 850, and calculation of probability estimates, as carried out in the probability estimation block 830, are done in the same manner as the encoder. The received bits are then fed into an arithmetic decoding engine 840 of the arithmetic decoder 820, where they are converted

to decoded bin values, using the calculated context and the current probability estimates of the bin values. Decoded bins are mapped to the values of the runs and levels in a bin-to-value mapping block 860.

CABAC Method as Used in H.26L

The details of the CABAC arithmetic coding method adopted for use in the high complexity profile of ITU-T recommendation H.26L will now be described in detail. According to H.26L TML8, the contexts for run and level values depend on the type of block being encoded and the bin number of the binarized level or run value. Different block types are defined according to the scanning mode (single/double) used to order the coefficient values, the component type (luminance/chrominance, AC/DC), or the coding mode (INTER/INTRA). However, for a given block type, the context depends only on the bin number. More specifically, according to H.26L TML8 four contexts are defined for level encoding. The first one is for the first bin, the second is for the second bin, while the third context is for the rest of the bins representing the magnitude of the level. The remaining context is used for the sign of the level. A similar approach is used to assign run values to contexts. For runs there are three contexts, the first one for the first bin, the second for the second bin and the third for all remaining bins. As run values are always equal to or greater than zero, there is no need for an additional context to represent sign information. Thus, for a block of a given type, the assignment of bins to contexts for transform coefficient bins (for both level and run encoding) can be summarised as follows:

```

if (bin_nr > MAX_BIN_VAL)
    bin_nr = MAX_BIN_VAL;
end
context = bin_nr

```

where `bin_nr` is the bin number and `context` is the context number. According to H.26L TML8, the value of `MAX_BIN_VAL` is set equal to 3, but in principle another `MAX_BIN_VAL` could be used instead.

A run-level pair is encoded as follows: The runs and levels are first classified according to block/coefficient type: scanning mode, coefficient type (DC/AC), and coding mode (INTER/INTRA or 16x16 INTRA). The levels and runs are then binarized by mapping them onto a sequence of bins and each bin is assigned to a context based on its bin number.

FIGS. 6a–6d illustrate this process in detail with reference to an exemplary 4x4 array of quantized DCT coefficients. It also demonstrates the adaptive nature of the CABAC method, by illustrating the way in which the statistical properties of run and level values for quantized DCT coefficients are tracked. The two-dimensional array of quantized DCT coefficient values is first zigzag scanned to produce a one-dimensional array of values, as indicated in FIG. 6a. The non-zero coefficient values in the one-dimensional array are then represented as pairs of run and level values. As previously explained, each level value represents the value of a non-zero quantized DCT coefficient, while the associated run value corresponds to the number of zero-valued coefficients preceding the coefficient in question. The run-level pairs derived from the exemplary array of quantized DCT coefficients are presented in FIG. 6b. In each pair the level value precedes the associated run value and a level value equal to zero is used as an end-of-block symbol to indicate that there are no more non-zero coefficient values in the block.

Next, each run and level value is converted into a binary value. According to H.26L TML8, the binarization scheme

15

used to convert the run and level values for quantized DCT transform coefficient values is identical to that shown above in Table 1. FIG. 6c shows the result of applying the binarization scheme presented in Table 1 to the run and level values in the exemplary array. FIG. 6c also shows the assignment of bins to contexts according to H.26L. As described above, only three contexts are used to describe the magnitudes of the run and level values. The first context corresponds to bin 1, the second to bin 2, while the third context comprises all the remaining bins. In FIG. 6c, the contexts are delineated by bold horizontal lines. By examining FIG. 6c it can be seen that the majority of level values are mapped to bins which are assigned to context 3, while the majority of run values are mapped to bins which are assigned to context 1.

The probability estimates for each assigned context are updated after the encoding of the bins. Probability estimates for the run and level contexts are updated independently. As previously described, the probability estimate for a given context represents the statistical characteristics of the bins assigned to the context in question. More specifically, the probability estimate describes the likelihood of a bin assigned to the context containing a 1 or a 0. FIG. 6d describes, in an exemplary manner, the way in which the probability estimates are updated for runs and levels. The figure illustrates the probability of a bin assigned to a given run or level context containing a 1 or a 0 before and after the runs and levels representing the 4x4 block of quantized DCT coefficients shown in FIG. 6a are binarized and assigned to contexts and encoded in the arithmetic encoder. FIG. 6d takes the form of a table, which records the occurrence of 1's and 0's in the bins assigned to each context. Thus, the probability estimate for a given context is given by:

probability of 0=no. of 0's/(no. of 0's+no. of 1's)

probability of 1=no. of 1's/(no. of 0's+no. of 1's)

In the figure it is assumed that the 4x4 block of quantized DCT coefficients shown in FIG. 6a is the first such block to be processed. This means that there are no previous occurrences of 1's and 0's to record in the table. To overcome this problem it is assumed that before the block is processed, each context has an equal probability of containing a 1 or a 0. This is indicated by entering identical values in the columns that record the occurrence of 0's and 1's. In FIG. 6d, 1's are used to initialize the probability estimate. Alternatively, a probability estimate derived from analysis of training data could be used to initialize the probability estimates for each context. The probability estimates are then updated by counting the number of 1's and 0's which occur in the bins of each context as the run and level values for the block of quantized DCT transform coefficients are binarized and assigned to contexts. The right-hand column of FIG. 6d shows the situation after processing the 4x4 block of quantized DCT shown in FIG. 6a.

Although the CABAC arithmetic encoding method adopted in the high complexity profile of ITU-T recommendation H.26L TML8 provides an improvement in data compression compared with the UVLC entropy coding method, it is still not optimal with regard to coding efficiency. It is therefore an object of the invention to provide a method and system for context-based arithmetic coding, wherein coding efficiency is further improved.

SUMMARY OF THE INVENTION

The present invention is based on the realization that when coding a given data symbol using context-based arithmetic coding, an improvement in coding efficiency can be achieved by using context models which take into

16

account the contexts to which other data symbols are assigned. With specific reference to the CABAC method used in the high complexity profile of H.26L TML8, the inventors of the present invention have determined that certain relationships exist between the run and level values associated with DCT transform coefficients. They have further determined that these relationships can be used to construct improved context models which enable the CABAC method to operate with improved coding efficiency when applied to the run and level values. In particular, the inventors have determined that consecutive level values exhibit a significant similarity. More specifically, within a given block of transform coefficients, the level of a coefficient to be encoded has, in general, a magnitude substantially similar to the level of the previously encoded coefficient. The inventors have also determined an inverse relationship between the level and run values. In particular, larger level values are more likely to be preceded by smaller run values. The converse is also true, namely smaller level values are likely to be preceded by larger runs. Consequently, the present invention proposes the creation of new context models for the coding of DCT transform coefficients, which take into account these relationships between level and run values.

In a first such context model, intended for implementation in a context-based arithmetic encoder, the context assigned to the bins of a binarized coefficient level value depends on the previously encoded coefficient level. In a second such context model, intended for implementation in a context-based arithmetic decoder, the context assigned to the bins of a binarized coefficient level value depends on the previously decoded coefficient level. In a third context model, implemented in either a context-based arithmetic encoder or a context-based arithmetic decoder, the context assigned to the bins of a binarized coefficient run value depends on the coefficient's level value.

The inventors have also determined that certain similarities exist between the transform coefficient values associated with different image blocks. These similarities are greater between image blocks which reside close to each other and tend to be strongest between immediately neighbouring image blocks. More specifically, the number N_c of non-zero transform coefficient values representing a particular image block tends to be similar to the number of non-zero transform coefficient values in an image block close to, or neighbouring, the image block in question. Thus, the present invention further introduces the concept of providing an indication of the number of non-zero transform coefficients for a transform coded image block and encoding this value using entropy coding. Furthermore, if context-based arithmetic coding is used to encode the N_c value, the inventors have determined that it is advantageous to assign the N_c value of a block to a context by taking into account the context assigned to the N_c value for at least one other transform coded image block. In this way the similarity between N_c values between image blocks which reside close to each other can be taken advantage of in the context-based arithmetic coding procedure. According to ITU-T recommendation H.26L TML8, the number of non-zero transform coefficients in an image block is not encoded. Instead, and as previously explained, an end-of-block (EOB) indication is provided. The EOB indication signals that the last run-level pair corresponding to a non-zero coefficient has been encoded. The inventors have determined that the proposed method, in which an explicit indication of the number of non-zero coefficients in a block is provided and coded using context-based arithmetic coding leads to an increase in the

coding efficiency compared with the method of providing an EOB indication as currently employed in H.26L TML8.

Although the motivation behind the present invention and its basic concepts have been presented in the context of video encoding/decoding and more specifically with respect to H.26L TML8, it should be appreciated that invention may be applied in other video coding systems and also to still image coding. In principle the invention can be applied in any image coding system in which block-based transform coding and context-based arithmetic coding are used.

According to a first aspect of the present invention there is provided a method of image coding, wherein an image is divided in an encoder into a plurality of blocks having a plurality of pixels, each pixel having a pixel value, and a transform coding operation is performed on a block of pixel values to produce a corresponding block of transform coefficient values. The block of transform coefficient values is scanned in a given scanning order to produce a scanned array of coefficient values arranged according to the scanning order wherein the coefficient values in the scanned array are represented by a plurality of number pairs, each having a first number indicative of a non-zero coefficient value and a second number indicative of a number of consecutive zero coefficient values preceding the non-zero coefficient value, and wherein the first and the second number are each assigned to one of a plurality of contexts indicative of the number pairs. The method comprises the step of assigning a further value to a context based on a third number indicative of a preceding value.

Preferably, the further value is the first number in one of the number and the preceding value is the first number of a preceding number pair.

Preferably, the step of assigning the first number in one of the number pairs to a context based on a third number indicative of the first number of a preceding number pair further takes into account the context to which the first number of the number pair is assigned.

Advantageously, the first number is equal to the magnitude of a non-zero coefficient value.

Preferably, the contexts are contexts of a context-based arithmetic coder.

Even more preferably, the contexts are contexts of a context-based binary arithmetic coder.

Advantageously, each of the first, second and third numbers is mapped to a set of bins, each of the bins having an associated bin number and being capable of taking one of either a first value or a second value.

Preferably, each of the first, second and third numbers is mapped to one of the set of bins, mapping of a number to a given one of the set of bins being indicated by assigning the value of the bin to the first value.

Preferably the first value is 1 and the second value is 0.

Preferably each of the set of bins is assigned to a context.

Advantageously, the step of assigning the first number in one of the number pairs to a context based on a third number indicative of the first number of a preceding number pair further taking into account the context to which the first number in the preceding number pair is assigned is implemented by examining the bin number of the bin to which the first number of the preceding number pair is mapped.

Advantageously, the method further comprises maintaining a probability estimate describing the statistical properties of each context.

Preferably, for each context, the probability estimate is indicative of the statistical likelihood of a number having a predetermined value being assigned to the context.

Preferably, for each context, the probability estimate is maintained by keeping a record of occurrences of the first value and the second value in the bins assigned to the context in question.

According to a second aspect of the present invention there is provided a method of image coding, wherein an image is divided in an encoder into a plurality of blocks having a plurality of pixels, each pixel having a pixel value, and a transform coding operation is performed on a block of pixel values to produce a corresponding block of transform coefficient values. The block of transform coefficient values is scanned in a given scanning order to produce a scanned array of coefficient values arranged according to the scanning order wherein the coefficient values in the scanned array are represented by (converted into) a plurality of number pairs, each having a first number indicative of a non-zero coefficient value and a second number indicative of a number of consecutive zero coefficient values preceding the non-zero coefficient value, and wherein the first and the second number are each assigned to one of a plurality of contexts indicative of the number pairs. The method comprises the step of assigning the second number in one of the number pairs to a context based on the first number of the number pair.

Preferably, the step of assigning the second number in one of the number pairs to a context based on the first number of the number pair further takes into account the context to which the second number in the one of the number pairs is assigned.

Advantageously, the first number is equal to the magnitude of a non-zero coefficient value.

Preferably, the contexts are contexts of a context-based arithmetic coder.

Even more preferably, the contexts are contexts of a context-based binary arithmetic coder.

Advantageously, each of the first and second numbers is mapped to a set of bins, each of the bins having an associated bin number and being capable of taking one of either a first value or a second value.

Preferably, each of the first and second numbers is mapped to one of the set of bins, mapping of a number to a given one of the set of bins being indicated by assigning the value of the bin to the first value.

Preferably the first value is 1 and the second value is 0.

Preferably each of the set of bins is assigned to a context.

Advantageously, the step of assigning the second number in one of the number pairs to a context based on the first number of the number pair further taking into account the context to which the second number in the one of the number pairs is assigned is implemented by examining the bin number of the bin to which the second number is mapped.

Advantageously, the method further comprises maintaining a probability estimate describing the statistical properties of each context.

Preferably, for each context, the probability estimate is indicative of the statistical likelihood of a number having a predetermined value being assigned to the context.

Preferably, for each context, the probability estimate is maintained by keeping a record of occurrences of the first value and the second value in the bins assigned to the context in question.

Preferably, the methods according to the first and the second aspect of the invention are both applied to a block of transform coefficient values.

According to a third aspect of the present invention there is provided a system for image coding, wherein an image is

divided in an encoder into a plurality of blocks having a plurality of pixels, each pixel having a pixel value, and a transform coding operation is performed on a block of pixels to produce a corresponding block of transform coefficient values. The block of transform coefficient values is scanned in a given scanning order to produce a scanned array of coefficient values arranged according to the scanning order wherein the coefficient values in the scanned array are represented by a plurality of number pairs, each having a first number indicative of a non-zero coefficient value and a second number indicative of a number of consecutive zero coefficient values preceding the non-zero coefficient value, and wherein the first and the second number are each assigned to one of a plurality of contexts indicative of the number pairs. The system comprises a mechanism, located in the encoder, for assigning the first number in one of the number pairs to a context based on a third number indicative of the first number of a preceding number pair.

According to a fourth aspect of the present invention there is provided a system for image coding, wherein an image is divided in an encoder into a plurality of blocks having a plurality of pixels, each pixel having a pixel value, and a transform coding operation is performed on a block of pixels to produce a corresponding block of transform coefficient values. The block of transform coefficient values is scanned in a given scanning order to produce a scanned array of coefficient values arranged according to the scanning order wherein the coefficient values in the scanned array are represented by a plurality of number pairs, each having a first number indicative of a non-zero coefficient value and a second number indicative of a number of consecutive zero coefficient values preceding the non-zero coefficient value, and wherein the first and the second number are each assigned to one of a plurality of contexts indicative of the number pairs. The system comprises a mechanism, located in the encoder, for assigning the second number in one of the number pairs to a context based on the first number of the number pair.

According to a fifth aspect of the present invention there is provided a method of image coding, wherein an image is divided in an encoder into a plurality of blocks having a plurality of pixels, each pixel having a pixel value, and a transform coding operation is performed on a block of pixels to produce a corresponding block of transform coefficient values. The method comprises the step of providing a number indicative of the number of non-zero coefficient values in the block of transform coefficient values and assigning the number to a context.

Advantageously, the step of assigning the number indicative of the number of non-zero transform coefficient values in the block of transform coefficient values takes into account the context to which another such number indicative of the number of non-zero transform coefficient values in another block of transform coefficient is assigned.

Advantageously, the block of transform values is scanned in a given scanning order to produce a scanned array of coefficient values arranged according to the scanning order wherein the coefficient values in the scanned array are represented by a plurality of number pairs, each having a first number indicative of a non-zero coefficient value and a second number indicative of the number of consecutive zero coefficient values preceding the non-zero coefficient value.

Preferably, an end-of-block indication, indicative of the last non-zero coefficient value in the scanned array of coefficient values is not provided.

Preferably, the first number is equal to the magnitude of a non-zero coefficient value minus 1.

Preferably, the methods according to the first, second and fifth aspects of the invention are each applied to a block of transform coefficient values.

According to a sixth aspect of the present invention there is provided a system for image coding, wherein an image is divided in an encoder into a plurality of blocks having a plurality of pixels, each pixel having a pixel value, and a transform coding operation is performed on each block of pixels to produce a corresponding block of transform coefficient values. The system comprises a mechanism, located in the encoder and responsive to the coefficient values, for providing a third number indicative of the number of non-zero coefficient values in the block of transform coefficient values and assigning the number to a context.

Advantageously, the system further comprises a mechanism for assigning the number indicative of the number of non-zero transform coefficient values in the block of transform coefficient values taking into account the context to which another such number indicative of the number of non-zero transform coefficient values in another block of transform coefficient is assigned.

According to a seventh aspect of the present invention there is provided a computer program comprising code for performing a method of image coding, in which an image is divided into a plurality of blocks having a plurality of pixels, each pixel having a pixel value, and a transform coding operation is performed on a block of pixel values to produce a corresponding block of transform coefficient values. The computer program also comprises code for scanning the block of transform coefficient values in a given scanning order to produce a scanned array of coefficient values arranged according to the scanning order wherein the coefficient values in the scanned array are represented by a plurality of number pairs, each having a first number indicative of a non-zero coefficient value and a second number indicative of a number of consecutive zero coefficient values preceding the non-zero coefficient value, and wherein the first and the second number are each assigned to one of a plurality of contexts indicative of the number pairs. The computer program also comprises code for assigning the first number in one of the number pairs to a context based on a third number indicative of the first number of a preceding number pair.

According to an eighth aspect of the present invention there is provided a computer program comprising code for performing a method of image coding, wherein an image is divided in an encoder into a plurality of blocks having a plurality of pixels, each pixel having a pixel value, and a transform coding operation is performed on a block of pixel values to produce a corresponding block of transform coefficient values. The computer program also comprises code for scanning the block of transform coefficient values in a given scanning order to produce a scanned array of coefficient values arranged according to the scanning order wherein the coefficient values in the scanned array are represented by a plurality of number pairs, each having a first number indicative of a non-zero coefficient value and a second number indicative of a number of consecutive zero coefficient values preceding the non-zero coefficient value, and wherein the first and the second number are each assigned to one of a plurality of contexts indicative of the number pairs. The computer program further comprises code for assigning the second number in one of the number pairs to a context based on the first number of the number pair.

According to a ninth aspect of the present invention there is provided a computer program comprising code for per-

forming a method of image coding, wherein an image is divided in an encoder into a plurality of blocks having a plurality of pixels, each pixel having a pixel value, and a transform coding operation is performed on a block of pixels to produce a corresponding block of transform coefficient values. The computer program further comprises code for providing a number indicative of the number of non-zero coefficient values in the block of transform coefficient values and assigning the number to a context.

Advantageously, the computer program further comprises code for assigning the number indicative of the number of non-zero transform coefficient values in the block of transform coefficient values taking into account the context to which another such number indicative of the number of non-zero transform coefficient values in another block of transform coefficient is assigned.

According to a tenth aspect of the invention there is provided a computer program according to the seventh, eighth and ninth aspects of the invention.

The present invention will become apparent upon reading the description taken in conjunction with FIGS. 7a to 12.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram illustrating the structure of an exemplary video encoder, which employs block-based transform coding and motion-compensated prediction.

FIG. 2 is a block diagram of an exemplary video decoder corresponding to the encoder of FIG. 1.

FIG. 3 is a diagrammatic representation showing an exemplary zig-zag scan.

FIG. 4 is a block diagram showing an encoder in a prior art context-based arithmetic coding scheme.

FIG. 5 is a block diagram showing a decoder in a prior art context-based arithmetic coding scheme.

FIG. 6a is a diagrammatic representation showing an exemplary two-dimensional array of quantized DCT coefficient values scanned in a zigzag manner.

FIG. 6b is a table showing the level and run values derived from the array of FIG. 6a.

FIG. 6c is a table showing the binarized level and run values resulted from the application of the binarization scheme of Table 1 to the level and run values of FIG. 6b.

FIG. 6d is a table showing a way in which the probability estimates are updated from runs and levels.

FIG. 7a is a table showing a way in which contexts are assigned to bins based on level values.

FIG. 7b is a table showing the way in which contexts are assigned to level values according to the first embodiment of the present invention.

FIG. 8a is a table showing a way in which contexts are assigned to bins based on run values.

FIG. 8b is a table showing the way in which contexts are assigned to run values according to the second embodiment of the present invention.

FIG. 9 is a block diagram illustrating an encoder in a context-based arithmetic coding scheme, according to the present invention.

FIG. 10 is a block diagram illustrating a decoder, according to the present invention.

FIG. 11 is a flowchart illustrating a method of image coding, according to the preferred embodiment of the present invention.

FIG. 12 is a flowchart illustrating a method of image coding, according to another embodiment of the present invention.

BEST MODE TO CARRY OUT THE INVENTION

Embodiments of the invention will now be discussed in detail. As described above, the present invention provides a number of related mechanisms by which an improvement in the coding efficiency (data compression) of a context-based arithmetic coder, can be attained. This improvement is achieved by using context models, which take into account the contexts to which other data symbols are assigned.

The first embodiment of the invention, described in detail in section 1.1 below, relates to a context-based binary arithmetic coder suitable for use in an image coding system such as that defined by ITU-T recommendation H.26L. In this embodiment, level values generated by run-level coding the quantized transform coefficients of a transform coded block of image pixels are assigned to contexts taking into account the level of another transform coefficient belonging to the same block.

The second embodiment of the invention, described in detail in section 1.2, also relates to a context-based binary arithmetic coder for an image coding system such as that defined by ITU-T recommendation H.26L. In the second embodiment, run values produced by run-level coding the quantized DCT transform coefficients of a transform coded block of image pixels are assigned to contexts taking into account the level value of the run-level pair to which the run value belongs.

The third embodiment of the invention is described in section 1.3 and also relates to a context-based arithmetic coder for an image coding system such as that defined by ITU-T recommendation H.26L. According to the third embodiment, the number of non-zero transform coefficients N_c for a transform coded image block is determined and assigned to a context taking into account the context assigned to the N_c value for at least one other transform coded image block.

A preferred embodiment of the invention combines the functionality of the three above-mentioned embodiments.

As explained earlier in the text, the high complexity profile of ITU-T recommendation H.26L TML8 employs a form of context-based arithmetic coding known as CABAC. In a video encoder implemented according to H.26L, the CABAC method is used to encode a variety of different types of information produced by the encoder, including the transform coefficients generated by transform coding blocks of image pixels (in INTRA-coding mode) or prediction error values (in INTER-coding mode). The two-dimensional array of transform coefficients produced by transform coding a block of image pixels is scanned according to a particular scanning mode to produce a one-dimensional array. Two such scanning modes are defined in H.26L. The first is known as 'single-scanning mode' while the other is referred to as 'double-scanning mode'. Whichever scanning mode is used, scanning of the transform coefficients converts the two-dimensional array of coefficient values into a one-dimensional array in which the coefficients are ordered in a predetermined manner. The ordered transform coefficient values in the one-dimensional array are converted run and level values. The last entry in the ordered one-dimensional array is an end-of-block symbol, which according to H.26L TML8, takes the form of a level value equal to zero. This indicates that the last non-zero coefficient value in the ordered array has been converted into a run-level pair.

The run and level values are converted to binary numbers (binarized), by mapping them to a series of bins, each of which can be assigned the value 0 or 1 (see Table 1). The

binarized run and level values are then assigned to contexts, a separate set of contexts being defined for the runs and the levels. According to H.26L TML8, for a given block type, the set of contexts defined for levels depends only on the bin number to which the levels are assigned. More specifically, according to H.26L TML8 four contexts are defined for level encoding. The first one is for the first bin, the second is for the second bin, while the third context is for the rest of the bins representing the magnitude of the level. The remaining context is used for the sign of the level. For runs there are three contexts, the first one for the first bin, the second for the second bin and the third for all remaining bins. As run values are always equal to or greater than zero, there is no need for an additional context to represent sign information.

1.1. Context Model for Levels

According to the first embodiment of the present invention, when assigning a binarized level value to a context, in addition to considering the bin to which the level value itself is mapped, the level value of the preceding run-level pair is also taken into account. In this context, the term 'preceding run-level pair' means the run-level pair corresponding to the preceding coefficient in the ordered one-dimensional array of coefficient values. The following pseudo code presents an exemplary procedure for assigning a context to a level value of a run-level pair, taking into account both the bin to which the level itself is mapped and the level value of the preceding run-level pair:

```

if (bin_nr > MAX_BIN_LEVEL)
    bin_nr = MAX_BIN_LEVEL;
end
if (prev_level > MAX_LEVEL)
    prev_level = MAX_LEVEL;
end
context = (bin_nr-1)*MAX_LEVEL + prev_level

```

In expression (2) prev_level is the (magnitude of the) level value of the previous run-level pair. prev_level is initialized to zero at the beginning of each block. In a double scanning mode, prev_level is initialized at the beginning of each scan, twice per block. Parameter MAX_BIN_LEVEL provides a means of controlling the way in which the bin number to which the level value is mapped affects the assignment of a context. More specifically, and in a manner similar to the present assignment of contexts according to H.26L TML8, MAX_BIN_LEVEL effectively defines a context to which all bin numbers greater than or equal to MAX_BIN_LEVEL are assigned. In an similar fashion parameter MAX_LEVEL provides a means of controlling the way in which the level value of the previous run-level pair affects the assignment of a context. FIGS. 7a and 7b illustrate the way in which contexts are assigned to level values according to the first embodiment of the invention by applying the pseudo code of expression (2) with MAX_BIN_LEVEL=3 and MAX_LEVEL=5. In principle, any combination of MAX_BIN_LEVEL and MAX_LEVEL can be used to define a set of contexts appropriate for the statistical characteristics of the level values to be coded.

1.2. Context Model for Runs

According to the second embodiment of the invention an approach is similar to that described in section 1.1 is used to assign run values to contexts. More specifically, when assigning a binarized run value to a context, in addition to considering the bin to which the run value itself is mapped, the level of the run-level pair to which the run value belongs is also taken into account. The following pseudo code presents an exemplary procedure for assigning a context to

a run value of a run-level pair, taking into account both the bin to which the run itself is mapped and the level value of the run-level pair to which the run value belongs:

```

if (bin_nr > MAX_BIN_RUN)
    bin_nr = MAX_BIN_RUN;
end
if (level > MAX_RUNL)
    level = MAX_RUNL;
end
context = (bin_nr-1)*MAX_RUNL + level

```

In expression (3) level is the magnitude of the level value of the run-level pair. Parameter MAX_BIN_RUN provides a means of controlling the way in which the bin number to which the run value is mapped affects the assignment of a context. More specifically, and in a manner similar to the present assignment of contexts according to H.26L TML8, MAX_BIN_RUN effectively defines a context to which all bin numbers greater than or equal to MAX_BIN_RUN are assigned. In an similar fashion parameter MAX_RUNL provides a means of controlling the way in which the level value of run-level pair affects the assignment of a context. FIGS. 8a and 8b illustrates the way in which contexts are assigned to level values according to the second embodiment of the invention by applying the pseudo code of expression (3) with MAX_BIN_RUN=3 and MAX_RUNL=4. In principle, any combination of MAX_BIN_RUN and MAX_RUNL can be used to define a set of contexts appropriate for the statistical characteristics of the run values to be coded.

1.3 Contexts for Number of Non-Zero Coefficients

The third embodiment of the invention relates in particular to the way in which an ordered array of transform coefficient values is converted into run and level values and the way in which the number of run-level pairs corresponding to an array of quantized transform coefficient values is signaled. More specifically, after a block of image pixels or prediction error values has been transform coded to form a two-dimensional array of transform coefficient values and each of the coefficient values has been quantized, the number of non-zero quantized coefficient values in the array is determined. A value, referred to as N_c , is assigned to that number and is used to signal explicitly the number of non-zero coefficient values in the array. Thus, according to this embodiment of the invention, an EOB symbol, for example a level value equal to zero, is no longer required.

The quantized transform coefficients are further scanned according to a predetermined scanning order to produce an ordered one-dimensional array. Alternatively, N_c may be determined after ordering the quantized coefficient values. Each of the non-zero quantized coefficients in the ordered array is then converted into a run-level pair. According to this embodiment of the invention, the level value of the run-level pair denotes the magnitude of the value of the quantized coefficient minus 1 and the run value corresponds to the number of consecutive zero-valued quantized coefficients preceding the coefficient in question. The level values are assigned to the magnitude of the value of the quantized coefficient minus 1 because a level value equal to zero is no longer used as an end-of-block indicator. This gives rise to a saving in the amount of data (e.g. number of bits) required to represent the level information.

The level and run values are then encoded using entropy coding, as is the N_c value. In a situation where a context-based arithmetic coding method such as the CABAC technique implemented in H.26L TML8 is used, the run and

level values may be encoded according to the first and/or second embodiments of the invention, as described above. Alternatively any other appropriate context models may be used for the run and level values. Additionally a separate context model is defined for N_c . According to this embodiment of the invention, the N_c value representing the number of non-zero quantized transform coefficients in a given block is first binarized by mapping it to a series of bins, each of which has a corresponding bin number. The context for N_c is then determined on the basis of the bin number to which N_c is mapped and the N_c of at least one other image block or macroblock which has already been assigned an N_c value. The following pseudo code presents an exemplary procedure for assigning a context to an N_c value, taking into account both the bin to which the N_c itself is mapped and the preceding N_c value:

```

if (bin_nr > MAX_BIN_Nc)
    bin_nr = MAX_BIN_Nc;
end
if (prev_nc > MAX_Nc)
    prev_nc = MAX_Nc;
end
context = (bin_nr-1)*MAX_nc + prev_nc

```

In expression (4) prev_nc is the previous N_c value.

When encoded level and run values for a given block of quantized transform coefficients are transmitted from an encoder to a decoder, the entropy coded N_c value is transmitted before the encoded run and level values. At the decoder, the N_c value is decoded, followed by the run-level pairs corresponding to the quantized transform coefficient values for the block in question. The value of +1 is added to each of the magnitude of the level values as they are decoded in order to compensate for the corresponding subtraction made at the encoder.

To demonstrate the improvement in coding efficiency using the method of image coding, according to the present invention, the average bitrate difference is calculated using results from QP=28, 24, 20, 16. Table 2 shows the bitrate reduction in percentage, as compared to TML-8, where MAX_LEVEL=5 and MAX_RUN=4. All frames are encoded as I-frames in CABAC mode. As shown in Table 2, the reduction in bitrate ranges from 0.95 to 4.74%. The improvement is more pronounced when the QP values are small.

TABLE 2

QP	Con- tainer	Fore- man	News	Silent	Tempete	Mobile	Paris
5	3.19	3.92	3.11	4.74	4.01	3.63	3.34
10	3.10	3.39	2.85	4.32	3.88	3.73	3.04
16	2.64	2.67	2.26	3.17	3.37	3.37	2.55
20	2.20	2.14	1.76	2.38	2.79	2.90	2.20
24	1.30	1.51	1.35	2.28	1.89	2.01	1.54
28	1.16	0.95	0.99	1.76	1.55	1.57	1.18
Ave. Bitrate	1.79	1.83	1.58	2.37	2.49	2.40	1.87
Diff.* (%)							

In Tables 2, the names appearing on the first row of the table are pictures used in Gisle Bjontegaard "Recommended Simulation Conditions for H.26L" (VCG-M75, ITU-T Video Coding Experts Group, Austin, Tex., USA, 2-4, Apr., 2001).

Referring now to FIG. 9, an encoder 10 in the transmit side, according to the present invention, includes a unit 16 for storing previous levels and runs. As shown in FIG. 9, the

run-level pairs 102 for a given block are provided to a mapping unit 12, which map the pairs to a sequence of bins, each bin having a value of 0 or 1. The location of the bin in the sequence representing a run-level pair is called a bin number. The bin numbers are represented by signals 104. Based on the signals 104 and a previously encoded level value 108 provided by unit 16, an assignment unit 14 assigns a context to a bin number. The contexts, denoted by signals 106, are provided to an adaptive arithmetic coder 20. The probability of occurrence of 1 and the probability of occurrence of 0 are estimated by a probability estimation module 22. Based on the probability estimates 120, an arithmetic encoding unit 24 encodes the bins. A feedback signal 124 is provided from the encoder 24 to the probability estimation module 22 to update the probability estimation. The encoded information is made into a bit-stream 122 to be conveyed to a decoder or stored in a storage device for later use.

Preferably, the encoder 10 also includes a unit 18, which is capable of providing the number, N_c , of non-zero coefficients in the block to the arithmetic encoder 20 before the run-level pairs are provided to the arithmetic encoder 20, so that N_c is also encoded and included into the bit-stream 122. N_c is represented by signals 110. By using N_c , there is no need to send an End-of-Block (EOB) symbol to the decoder. In prior art, the level value of 0 is used for the EOB symbol. More specifically, N_c is found after transform and quantization and it is encoded with entropy encoding. It should be noted that with the number of non-zero coefficients known, it is no longer necessary to use the 0-level value to signal the end of the block. Thus, it is possible to modify the level value by subtracting the value of the quantized coefficient by 1.

On the receive side, as shown in FIG. 10, a decoder 50 is used to receive the bit-stream 122 provided by the encoder 10. The received bit-stream, which represents arithmetic coded data symbols, is denoted by reference numeral 202. Initially, based on the previously decoded symbols, a context is calculated in a context assignment block 66 and the probability estimates of the bin values are updated in a probability estimation block 62. The previously decoded symbols based on which the probability estimates are updated are denoted by reference numeral 205. Context assignment, as carried out in the context assignment block 66, and calculation of probability estimates, as carried out in the probability estimation block 62, are similar to those in the encoder 10. The received bits 202 are then fed into an arithmetic decoding engine 64 in an arithmetic coder 60, where they are converted into decoded bin values 206, using the calculated context and the current probability estimates of the bin values 204. Decoded bins 208 are mapped to the values of the runs and levels in block 68. If the number, N_c , of non-zero coefficients in the block is encoded in the encoder 10 and provided in the received bit-stream 202, then a signal 214 is provided to the bin-to-value mapping module 68 whereby the quantized coefficient is restored by adding to the level value by 1.

FIG. 11 is a flowchart illustrating a method of image coding, according to the preferred embodiment of the present invention. As shown, the method 500 starts at step 510 where an image is received by an encoder. The received image is divided into a plurality of blocks at step 520. Each block is scanned at step 530 and the levels and runs of the quantized coefficients in the block are obtained at step 540. In contrast to prior art coding schemes, the present invention also uses the previous levels in the assignment of contexts at step 550. In particular, the assignment of contexts to the bins representing the level values of the quantized coefficients

takes into account the value of the previously encoded level, as described in section 1.1. Likewise, the assignment of contexts to the bins representing the run values of the quantized coefficients takes into account the level value, as described in section 1.2. The assigned contexts are conveyed to an arithmetic coder for encoding at step 560. Additionally, N_c , or the number of non-zero quantized coefficients, is determined during or after the block is scanned at step 530 and N_c is encoded at step 560 prior to N_c and the contexts being provided to a decoder, as described in section 1.3.

Alternatively, the image coding method can be improved solely by conveying signals indicative of N_c to the receive side, without considering the value of the previously encoded level or run when the contexts are assigned, as shown in FIG. 11. As shown in FIG. 12, instead of obtaining the previously encoded levels and runs at step 540 for assigning the contexts at step 550, N_c is obtained and provided at step 542. N_c is conveyed, before the contexts assigned at step 550 are sent, to an arithmetic coder for encoding at step 560. By sending N_c , there is no need to send the EOB symbol to the decoder.

Although the invention has been described with respect to a preferred embodiment thereof, it will be understood by those skilled in the art that the foregoing and various other changes, omissions and deviations in the form and detail thereof may be made without departing from the scope of this invention.

What is claimed is:

1. A method of image coding, wherein an image is divided in an encoder into a plurality of blocks having a plurality of pixels, each pixel having a pixel value, and a transform coding operation is performed on a block of pixels to produce a corresponding block of transform coefficient values, the block of transform coefficient values being scanned in a given scanning order to produce a scanned array of coefficient values arranged according to the scanning order, the method comprising the steps of:

representing the coefficient values in the scanned array by a plurality of number pairs, each of said number pairs having a first number and a second number; and assigning the first numbers to one of a plurality of contexts representative of the first numbers such that the first number of a first number pair is assigned to a context at least partly in dependence on a first number of a second number pair.

2. A method according to claim 1, wherein the first number of the first number pair is assigned to a context at least partly in dependence on the first number of a preceding number pair in the scanning order.

3. A method according to claim 1, wherein said plurality of contexts are contexts of a context-based arithmetic coder.

4. A method according to claim 1, wherein said plurality of contexts are contexts of a context-based binary arithmetic coder.

5. A method according to claim 4, further comprising the step of:

mapping the first numbers of said number pairs to a set of bins, each of which can be assigned one of either a first or a second value.

6. A method according to claim 5, wherein mapping of a number to a given one of said set of bins is indicated by assigning the bin to the first value.

7. A method according to claim 5, wherein the first value is 1 and the second value is 0.

8. A method according to claim 5, wherein the first number of a number pairs is assigned to a context at least partly in dependence on the bin to which the first number of said number pair is mapped.

9. A method according to claim 1, wherein said first number is indicative of a non-zero coefficient value and said second number is indicative of a number of consecutive zero coefficient values preceding said non-zero coefficient value.

10. A method of image coding, wherein an image is divided in an encoder into a plurality of blocks having a plurality of pixels, each pixel having a pixel value, and a transform coding operation is performed on a block of pixels to produce a corresponding block of transform coefficient values, the block of transform coefficient values is scanned in a given scanning order to produce a scanned array of coefficient values arranged according to the scanning order, the method comprising the steps of:

representing the coefficient values in the scanned array by a plurality of number pairs, each of said number pairs having a first number and a second number; and assigning the second numbers to one of a plurality of contexts representative of the second numbers such that the second number of a number pair is assigned to a context at least partly in dependence on the first number of the number pair.

11. A method according to claim 10, wherein said plurality of contexts are contexts of a context-based arithmetic coder.

12. A method according to claim 10, wherein said plurality of contexts are contexts of a context-based binary arithmetic coder.

13. A method according to claim 12, further comprising the step of:

mapping the second numbers of said number pairs to a set of bins, each of which can be assigned one of either a first value or a second value.

14. A method according to claim 13, wherein mapping of a number to a given one of said set of bins is indicated by assigning the bin to the first value.

15. A method according to claim 13, wherein the first value is 1 and the second value is 0.

16. A method according to claim 13, wherein the second number of a number pair is assigned to a context at least partly in dependence on the bin to which the second number of the number pair is mapped.

17. A method according to claim 10, wherein said first number is indicative of a non-zero coefficient value and said second number is indicative of a number of consecutive zero coefficient values preceding said non-zero coefficient value.

18. A system for image coding comprising:

an input for receiving an image as a plurality of blocks having a plurality of pixels, each pixel having a pixel value;

a transform coder for performing a transform coding operation on a block of pixels to produce a corresponding block of transform coefficient values;

a scanner for scanning the block of transform coefficient values in a given scanning order to produce a scanned array of coefficient values, arranged according to the scanning order;

a run-level coder for representing the coefficient values in the scanned array by a plurality of number pairs, said number pairs having a first number and a second number; and

a context-based coder for assigning the first numbers to one of a plurality of contexts representative of the first numbers and operative to assign the first number of a first number pair to a context at least partly in dependence on a first number of a second number pair.

19. A system according to claim 18, wherein said context-based coder is a context-based arithmetic coder.

29

20. A system according to claim 18, wherein said context-based coder is a context-based binary arithmetic coder.

21. A system according to claim 20, wherein the context-based coder is operative to map the first numbers of said number pairs to a set of bins, each of which can be assigned to one of either a first or a second value. 5

22. A system according to claim 21, wherein the context-based coder is operative to map a number to a given one of said set of bins by assigning the bin to the first value.

23. A system according to claim 21, wherein the first value is 1 and the second value is 0. 10

24. A system according to claim 21, wherein the context-based coder is operative to assign the first number of a number pair to a context at least partly in dependence on the bin to which the first number of said number pair is mapped. 15

25. A system according to claim 18, wherein the context-based coder is operative to assign the first number of the first number pair to a context at least partly in dependence on the first number of a preceding number pair in the scanning order. 20

26. A system according to claim 18, wherein said first number is indicative of a non-zero coefficient value and said second number is indicative of a number of consecutive zero coefficient values preceding said non-zero coefficient value.

27. A system for image coding comprising: 25

an input for receiving an image as a plurality of blocks having a plurality of pixels, each pixel having a pixel value;

a transform coder for performing a transform coding operation on a block of pixels to produce a corresponding block of transform coefficient values; 30

a scanner for scanning the block of transform coefficient values in a given scanning order to produce a scanned array of coefficient values arranged according to the scanning order; 35

a run-length coder for representing the coefficient values in the scanned array by a plurality of number pairs, said number pairs having a first number and a second number; and 40

a context-based coder for assigning the second numbers to one of a plurality of contexts representative of the second numbers, and operative to assign the second number of a number pair to a context at least partly in dependence on the first number of the number pair. 45

28. A system according to claim 27, wherein said context-based coder is a context-based arithmetic coder.

29. A system according to claim 27, wherein said context-based coder is a context-based binary arithmetic coder.

30. A system according to claim 29, wherein the context-based coder is operative to map the second numbers of said number pairs to a set of bins, each of which can be assigned to one of either a first or a second value. 50

31. A system according to claim 30, wherein the context-based coder is operative to map a number to a given one of said set of bins by assigning the bin to the first value. 55

32. A system according to claim 30, wherein the first value is 1 and the second value is 0.

33. A system according to claim 30, wherein the context-based coder is operative to assign the second number of a number pair to a context at least partly in dependence on the bin to which the second number of the number pair is mapped. 60

34. A system according to claim 27, wherein said first number is indicative of a non-zero coefficient value and said second number is indicative of a number of consecutive zero coefficient values preceding said non-zero coefficient value. 65

30

35. A computer program embodied in a computer readable medium comprising:

machine-readable program code for dividing an image into a plurality of blocks having a plurality of pixels, each pixel having a pixel value;

machine-readable program code for performing a transform coding operation on a block of pixel values to produce a corresponding block of transform coefficient values;

machine-readable program code for scanning the block of transform coefficient values in a given scanning order to produce a scanned array of coefficient values arranged according to the scanning order;

machine-readable program code for representing the coefficient values in the scanned array by a plurality of number pairs, said number pairs having a first number and a second number; and

machine-readable program code for assigning the first numbers to one of a plurality of contexts representative of the first numbers such that the first number of a first number pair is assigned to a context at least partly in dependence on a first number of second number pair.

36. A computer program embodied in a computer readable medium comprising:

machine-readable program code for dividing an image into a plurality of blocks having a plurality of pixels, each pixel having a pixel value;

machine-readable program code for performing a transform coding operation on a block of pixel values to produce a corresponding block of transform coefficient values;

machine-readable program code for scanning the block of transform coefficient values in a given scanning order to produce a scanned array of coefficient values arranged according to the scanning order; 35

machine-readable program code for representing the coefficient values in the scanned array by a plurality of number pairs, said number pairs having a first number and a second number; and 40

machine-readable code for assigning the second numbers to one of a plurality of contexts representative of the second numbers such that the second number of a number pair is assigned to a context at least partly in dependence on the first number of the number pair.

37. A method of context-based arithmetic encoding in which an array of data symbols is represented with a code-word, the method comprising:

representing the data symbols in said array as number pairs comprising a first number and a second number; and

assigning the first number of a first number pair to a context selected from a plurality of contexts representative of the first numbers at least partly in dependence on a first number of a second number pair.

38. A method of context-based arithmetic encoding in which an array of data symbols is represented with a code-word, the method comprising:

representing the data symbols in said array as number pairs comprising a first number and a second number;

assigning the first number of a first number pair to a context selected from a plurality of contexts representative of the second numbers at least partly in dependence on the first number of the number pair.

39. A context-based arithmetic encoder operative to represent an array of data symbols with a code-word and comprising:

31

a run-level coder for representing the data symbols in said array as number pairs comprising a first number and a second number;

a context assigner for assigning the first number of a first number pair to a context selected from a plurality of contexts representative of the first numbers at least partly in dependence on a first number of a second number pair. 5

40. A context-based arithmetic encoder operative to represent an array of data symbols with a code-word and comprising: 10

32

a run-level coder for representing the data symbols in said array as number pairs comprising a first number and a second number;

a context assigner for assigning the second number of a number pair to a context selected from a plurality of contexts representative of the second numbers at least partly in dependence on the first number of the number pair.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 6,856,701 B2
DATED : February 15, 2005
INVENTOR(S) : Karczewicz et al.

Page 1 of 1

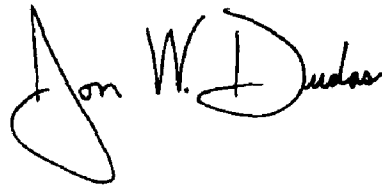
It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 30,

Line 61, "first number of a first number pair" should be -- second number of a number pair --.

Signed and Sealed this

Twenty-first Day of June, 2005

A handwritten signature in black ink, reading "Jon W. Dudas". The signature is stylized, with a large, looped initial "J" and a cursive "Dudas".

JON W. DUDAS
Director of the United States Patent and Trademark Office

EXHIBIT 11



US009800891B2

(12) **United States Patent**
Kalevo et al.

(10) **Patent No.:** **US 9,800,891 B2**
(45) **Date of Patent:** **Oct. 24, 2017**

(54) **METHOD AND ASSOCIATED DEVICE FOR
FILTERING DIGITAL VIDEO IMAGES**

(56) **References Cited**

U.S. PATENT DOCUMENTS

- (75) Inventors: **Ossi Kalevo**, Toijala (FI); **Emre Aksu**,
Tampere (FI); **Marta Karczewicz**,
Irving, TX (US)
- (73) Assignee: **Nokia Technologies Oy**, Espoo (FI)
- (*) Notice: Subject to any disclaimer, the term of this
patent is extended or adjusted under 35
U.S.C. 154(b) by 0 days.

5,218,649	A *	6/1993	Kundu et al.	382/180
5,225,904	A *	7/1993	Golin et al.	375/240.12
5,768,438	A *	6/1998	Etoh	
5,812,702	A *	9/1998	Kundu	382/260
5,896,176	A *	4/1999	Das et al.	375/240.15
5,912,706	A *	6/1999	Kikuchi et al.	375/240.13
6,181,802	B1 *	1/2001	Todd	382/100
6,240,135	B1 *	5/2001	Kim	375/240.01
RE37,668	E *	4/2002	Etoh	382/251
6,608,865	B1 *	8/2003	Itoh	375/240.08

FOREIGN PATENT DOCUMENTS

(21) Appl. No.: **09/766,035**

(22) Filed: **Jan. 19, 2001**

(65) **Prior Publication Data**

US 2001/0017944 A1 Aug. 30, 2001

DE	196 04 050	A1	2/1997
EP	0859518	A1	8/1998
EP	0 881 837	A1	12/1998
EP	0884911	A1	12/1998
EP	0 961 229	A2	12/1999

(Continued)

OTHER PUBLICATIONS

(30) **Foreign Application Priority Data**

Jan. 20, 2000 (FI) 20000120

Kim et al., "A Deblocking Filter with Two Separate Modes in
Block-Based Video Coding", IEEE Transactions on Circuits and
Systems for Video Technology, vol. 9, No. 1, pp. 156-160, Feb.
1999.*

(Continued)

(51) **Int. Cl.**

H04N 7/12 (2006.01)

H04N 19/527 (2014.01)

H04N 19/86 (2014.01)

(52) **U.S. Cl.**

CPC **H04N 19/527** (2014.11); **H04N 19/86**
(2014.11)

(58) **Field of Classification Search**

CPC H04N 19/0003; H04N 19/82; H04N 7/12
USPC 375/240.24-240.27, 240.29, 240.01,
375/240.08; 348/425.1-425.2, 420.1,
348/241, 245-246, 252, 533, 606-608,
348/620, 625, 627-630; 382/254-275

See application file for complete search history.

Primary Examiner — Young Lee

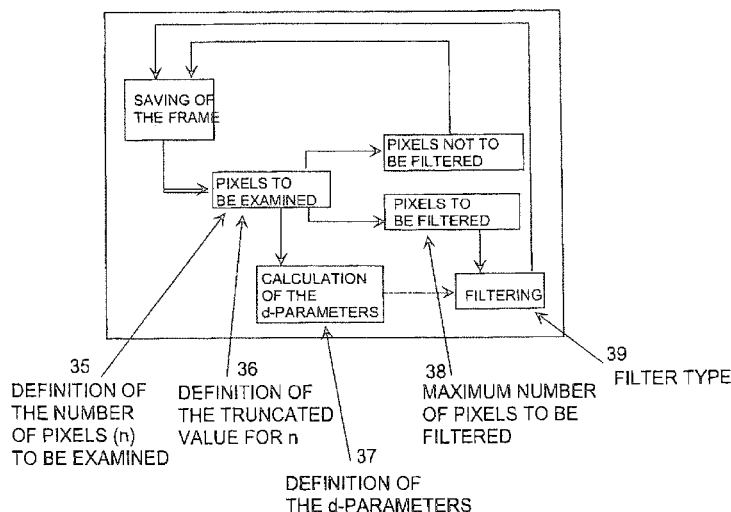
(74) *Attorney, Agent, or Firm* — Alston & Bird LLP

(57)

ABSTRACT

A method for reducing visual artifacts in a frame of a digital
video signal, which is coded by blocks and then decoded,
includes defining a block type is according to the coding
method for a block selected according to a predetermined set
of coding types. In the method filtering is performed to
reduce visual artifacts due to a block boundary. The filtering
performed depends on block types of the frame in the
environment of the block boundary.

41 Claims, 6 Drawing Sheets



(56)

References Cited

FOREIGN PATENT DOCUMENTS

GB	2 329 090 A	3/1999
GB	2329090	10/1999
RU	2042282 C1	8/1995
WO	WO 98/41025	9/1998
WO	WO 00/49809	8/2000

OTHER PUBLICATIONS

Australian Search Report and Written Opinion for Application SG
200405663-6 mailed Feb. 2, 2006.

Office Action for European Application No. EP 01 902 442.1 dated
Apr. 14, 2009.

International Search Report for Application No. PCT/FI01/00049
datd Apr. 17, 2001.

Decision on Grant for RU 2042282, Nov. 20, 2008, please see p. 15.

* cited by examiner

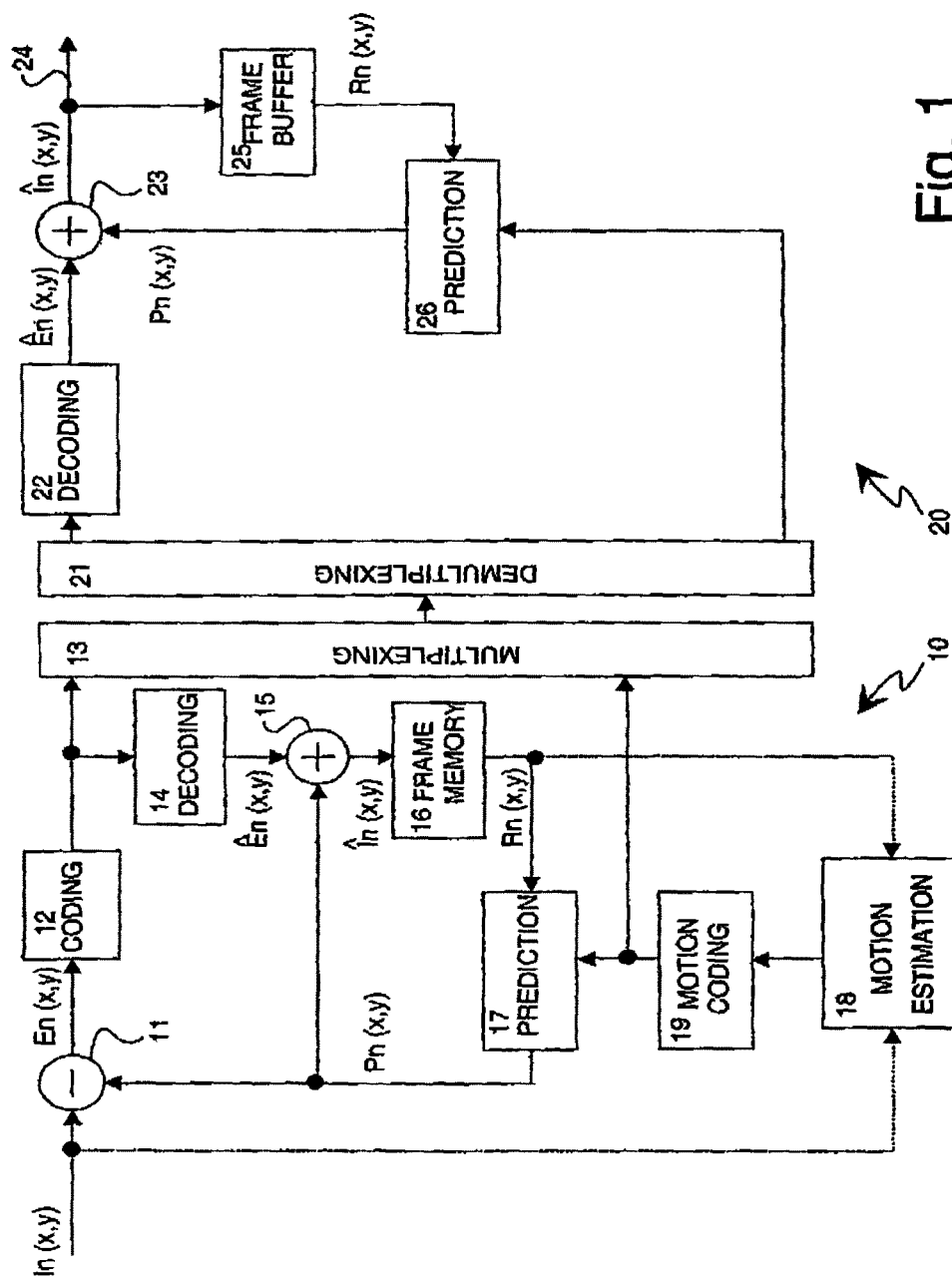


Fig. 1
PRIOR ART

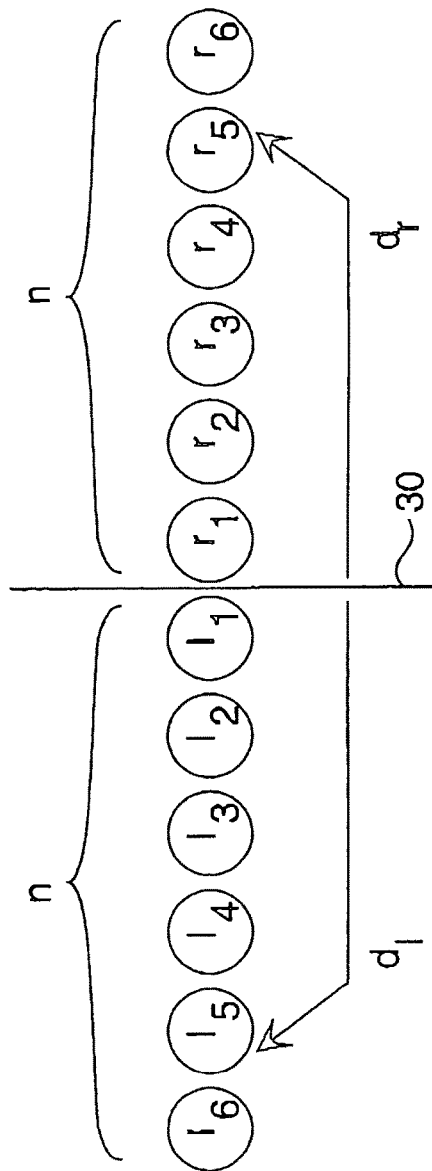
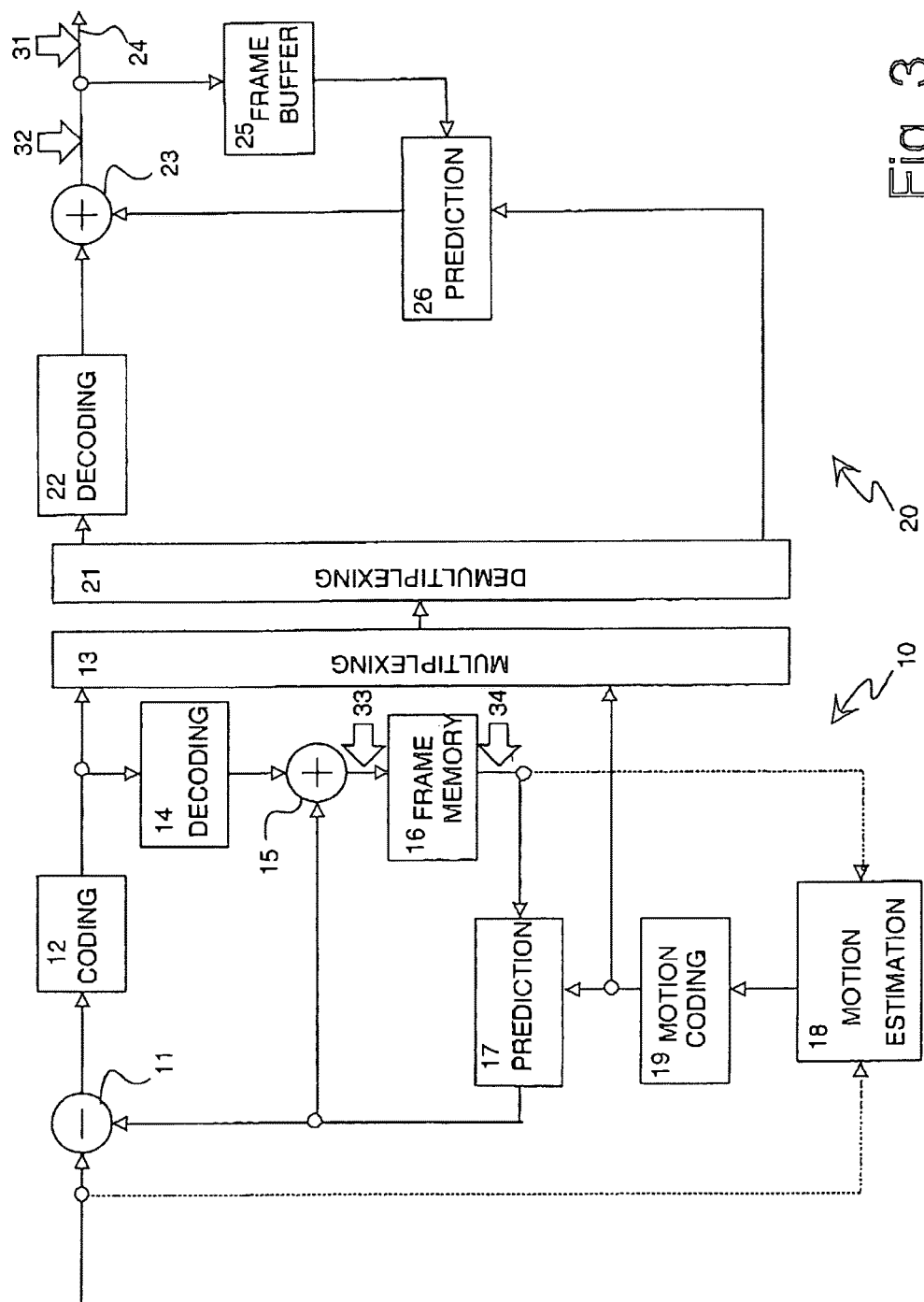


Fig. 2



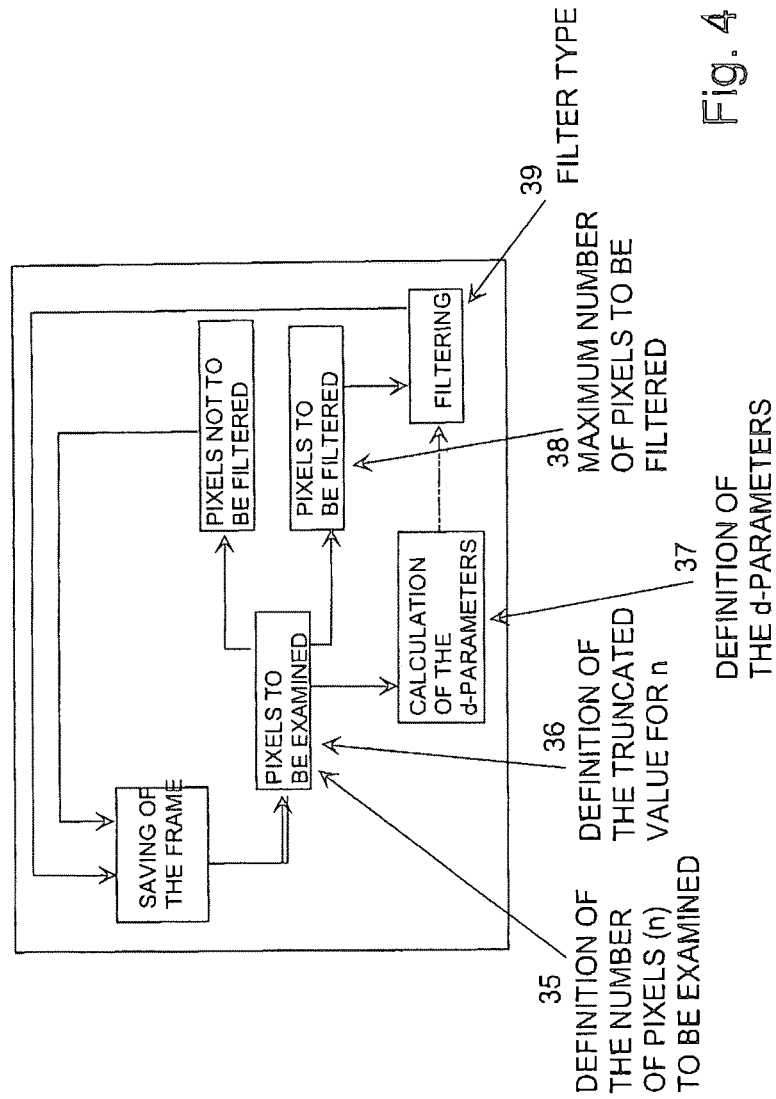


Fig. 4

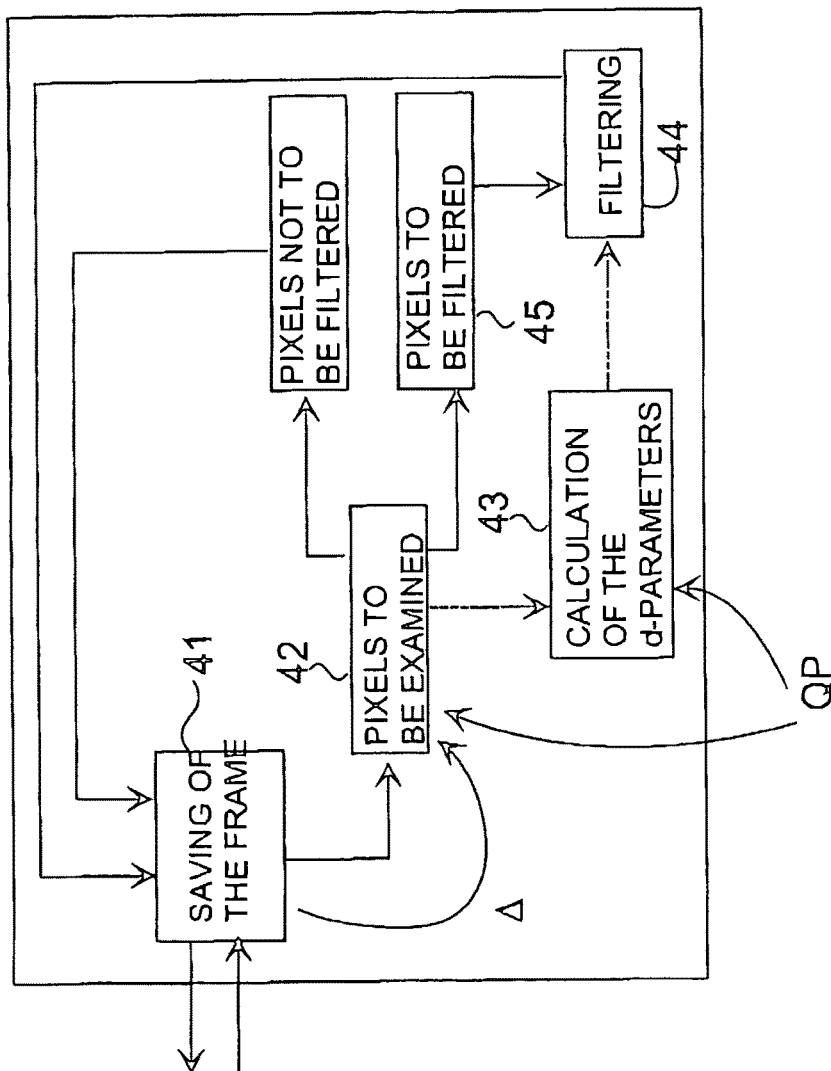


Fig. 5

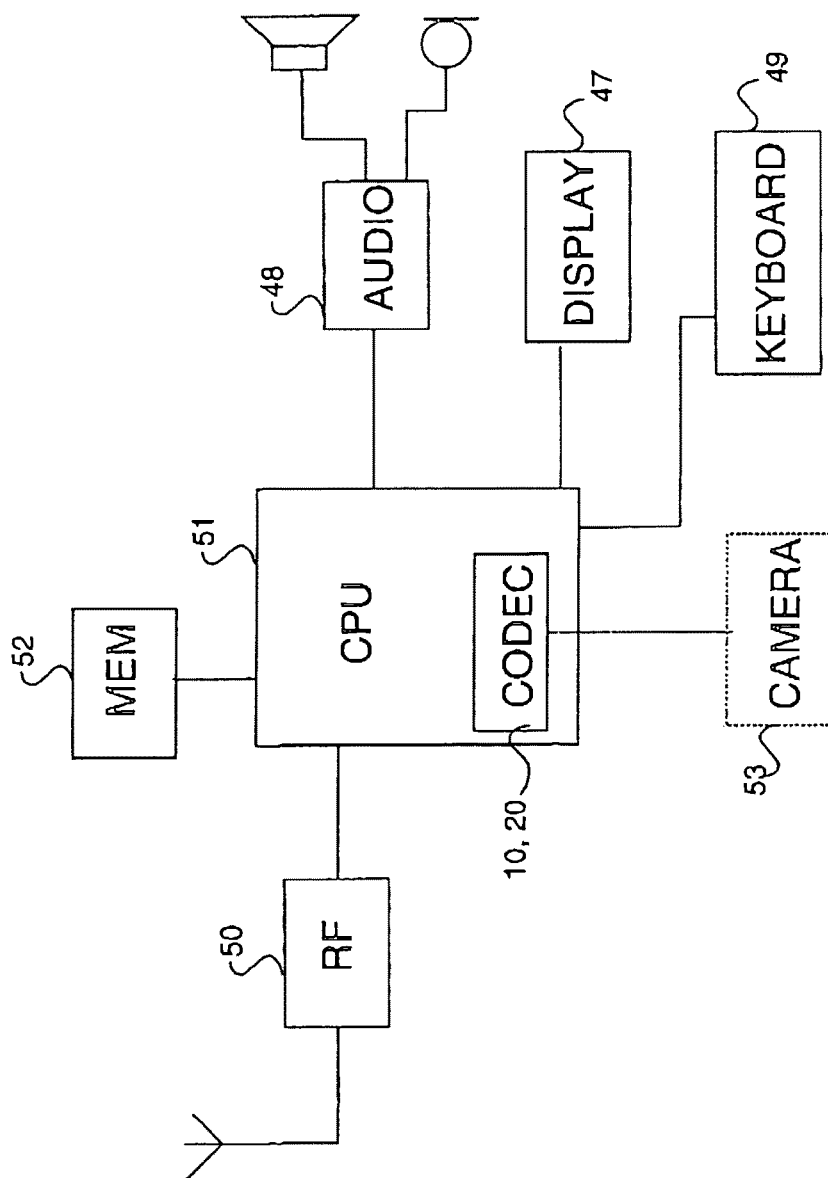


Fig. 6

METHOD AND ASSOCIATED DEVICE FOR FILTERING DIGITAL VIDEO IMAGES

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to a method for reducing visual artefacts in a frame of a digital video signal, which is coded by blocks and then decoded, a block type being defined according to the coding method for a block selected from a predetermined set of coding types, in which filtering is performed to reduce visual artefacts due to a block boundary. The present invention also relates to a device for reducing visual artefacts in a frame of a digital video signal, which is coded by blocks and then decoded, a block type being defined according to the coding method for a block selected according to a predetermined set of coding types, the device comprising a filter for reducing visual artefacts due to a block boundary. Furthermore, the present invention relates to a video encoder comprising means for coding and decoding a digital video signal by blocks, a block type being defined according to the coding method for a block selected according to a predetermined set of coding types, which encoder comprises a filter for reducing visual artefacts due to a block boundary. The present invention also relates to a video decoder comprising means for reducing visual artefacts in a frame of a digital video signal, which is coded by blocks and then decoded, a block type being defined according to the coding method for a block selected according to a predetermined set of coding types, which video decoder comprises a filter for reducing visual artefacts due to a block boundary. The present invention also relates to a video codec comprising means for coding and decoding a digital video signal by blocks, a block type being defined according to the coding method for a block selected according to a predetermined set of coding types, which video codec comprises a filter for reducing visual artefacts due to a block boundary. The present invention also relates to a mobile terminal comprising a video codec, which comprises means for coding and decoding a digital video signal by blocks, a block type being defined according to the coding method for a block selected according to a predetermined set of coding types, which video codec comprises a filter for reducing visual artefacts due to a block boundary. The present invention further relates to a storage medium for storing a software program comprising machine executable steps for coding and decoding a digital video signal by blocks, a block type being defined according to the coding method for a block selected according to a predetermined set of coding types, for reducing visual artefacts due to a block boundary by filtering.

2. Brief Description of Related Developments

A transmission system like that shown in FIG. 1 is generally used for transferring a digital video image in compressed form. The video image is formed of sequential frames. In some prior art video transmission systems, for example those according to the ITU-T H.261/H.263 recommendations, at least three frame types are defined: an I-frame (Intra), a P-frame (Predicted or Inter), and a B-frame (Bi-directional). The I-frame is generated solely on the basis of information contained in the image itself, wherein at the receiving end, this I-frame can be used to form the entire image. P-frames are formed on the basis of a preceding I-frame or P-frame, wherein at the receiving stage a preceding I-frame or P-frame is correspondingly used together with the received P-frame in order to reconstruct the image. In the composition of P-frames, for instance, motion compensation

is used to compress the quantity of information. B-frames are formed on the basis of one or more preceding P-frames or I-frames and/or one or more following P- or I-frames.

The frames are further divided into blocks. One or more such blocks forms a block region. There can generally be four different region types: Intra region, copy region, coded region, and not-coded region. An intra region is a block region in which the blocks are coded independently without reference to any other frame. A copy region consists of blocks which are obtained by copying the content of the reference frame into exactly the same location without any motion compensated prediction. A coded region consists of blocks which are obtained using motion compensated prediction and prediction error coding. The prediction error is a difference between the pixel values of the actual frame and a reconstructed frame which is formed in the coding/decoding system using motion compensated prediction, as will be described in further detail later in the text. The prediction error is coded and sent to a receiver. A not-coded region is obtained using motion compensated prediction only. In fact the not-coded region is equivalent to a copy region if the motion information equals 0. All the block regions of one frame are not necessarily similar types, but one frame can comprise block regions which are of different types.

Referring to FIG. 1, which illustrates a typical encoding and decoding system (codec) used, for example, in the transmission of digital video, a current video frame to be coded comes to the transmission system 10 as input data $I_n(x,y)$. The input data $I_n(x,y)$ typically takes the form of pixel value information. In the differential summer 11 it is transformed into a prediction error frame $E_n(x,y)$ by subtracting from it a prediction frame $P_n(x,y)$ formed on the basis of a previous image. The prediction error frame is coded in block 12 in a manner described hereinafter, and the coded prediction error frame is directed to a multiplexer 13. To form a new prediction frame, the coded prediction error frame is also directed to a decoder 14, which produces a decoded prediction error frame $\hat{E}_n(x,y)$ which is summed in a summer 15 with the prediction frame $P_n(x,y)$, resulting in a decoded frame $\hat{I}_n(x,y)$. The decoded frame is saved in a frame memory 16. To code the next frame, the frame saved in the frame memory 16 is read as a reference frame $R_n(x,y)$ and is transformed into a new prediction frame $P_n(x,y)$ in a motion compensation and prediction block 17, according to the formula:

$$P_n(x,y) = R_n[x + Dx(x,y), y + Dy(x,y)] \quad (1)$$

The pair of numbers $[Dx(x,y), Dy(x,y)]$ is called the motion vector of the pixel at location (x,y) and the numbers $Dx(x,y)$ and $Dy(x,y)$ are the horizontal and vertical shifts of the pixel. They are calculated in a motion estimation block 18. The set of motion vectors $[Dx(\bullet), Dy(\bullet)]$ consisting of all motion vectors related to the pixels of the frame to be compressed is also coded using a motion model comprising basis functions and coefficients. The basis functions are known to both the encoder and the decoder. The coefficient values are coded and directed to the multiplexer 13, which multiplexes them into the same data stream with the coded prediction error frame for sending to a receiver. In this way the amount of information to be transmitted is dramatically reduced.

Some frames can be partly, or entirely, so difficult to predict that it is not practical to use motion compensated prediction when coding them. These frames or parts of frames are coded using intra-coding without prediction, and therefore it is not necessary to send motion vector information relating to them to the receiver.

In the receiver system 20, a demultiplexer 21 separates the coded prediction error frames and the motion information transmitted by the motion vectors and directs the coded prediction error frames to a decoder 22. The decoder 22 produces a decoded prediction error frame $\hat{E}_n(x,y)$, which is summed in a summer 23 with the prediction frame $P_n(x,y)$ formed on the basis of a previous frame, resulting in a decoded frame $\hat{I}_n(x,y)$. The decoded frame is directed to an output 24 of the decoder and at the same time saved in a frame memory 25. When decoding the next frame, the frame saved in the frame memory is read as a reference frame and transformed into a new prediction frame in the motion compensation and prediction block 26, according to formula (1) presented above.

The coding method applied in block 12 to the coding of the prediction error frame or to the intra-coding of a frame or part of a P-frame to be sent without prediction, is generally based on a transformation, the most common of which is the Discrete Cosine Transformation, DCT. The frame is divided into adjacent blocks having a size of e.g. 8x8 pixels. In coding and decoding, the blocks are processed independent of one another. The transformation is calculated for the block to be coded, resulting in a series of terms. The coefficients of these terms are quantized on a discrete scale in order that they can be processed digitally. Quantization causes rounding errors, which can become visible in an image reconstructed from blocks, so that there is a discontinuity of pixel values at the boundary between two adjacent blocks. Because a certain decoded frame is used for calculating the prediction frame for subsequent predicted frames, these errors can be propagated in sequential frames, thus causing visible edges in the image reproduced by the receiver. Image errors of this type are called blocking artefacts.

Some prior art methods are known for removing blocking artefacts. These methods are characterized by the following features:

- determining which pixel/pixels require value correction in order to remove the blocking artefact,
- determining a suitable low-pass filtering for each pixel to be corrected, based on the values of other pixels contained by a filtering window placed around the pixel,
- calculating a new value for the pixel to be corrected, and rounding the new value to the closest digitized pixel value.

Factors that influence the selection of a filter and the decision whether to use filtering can be, for example, the difference between the values of pixels across the block boundary, the size of the quantization step of the coefficients received as the transformation result, and the difference of the pixel values on different sides of the pixel being processed.

It has been found that prior art methods tend to remove lines that belong to real features of the image. On the other hand, prior art methods are not always capable of removing all blocking artefacts.

SUMMARY OF THE INVENTION

An objective of the present invention is to present a new kind of filtering arrangement for reducing blocking artefacts. The invention also has the objective that the method and associated device operate more reliably and efficiently than prior art solutions.

The method according to the invention adjusts filtering parameters according to the type of blocks whose boundary

is to be filtered. Different filtering parameters are chosen according to the type of block on either side of the boundary in order to yield an improved filtering result.

The objectives of the invention are achieved by adapting the selection of pixels for filtering and the filtering process more flexibly than before to the features of the frame and the environment of the filtering point and by taking into account the nature/type of the blocks to be filtered.

According to a first aspect of the invention, there is provided a method for reducing visual artefacts in a frame that is coded by blocks, characterized in that the filtering performed on the block boundary depends on block types of the frame in the environment of the block boundary.

According to a second aspect of the invention, there is provided a device for implementing the method according to the invention. The device according to the invention is characterized in that the filter is arranged to operate adaptively according to the block types of the frame in the environment of the block boundary.

According to a third aspect of the invention, there is provided an encoder characterized in that the filter is arranged to operate adaptively according to the block types of the frame in the environment of the block boundary.

According to a fourth aspect of the invention, there is provided a decoder characterized in that the filter is arranged to operate adaptively according to the block types of the frame in the environment of the block boundary.

According to a fifth aspect of the invention, there is provided a codec characterized in that the filter is arranged to operate adaptively according to the block types of the frame in the environment of the block boundary.

According to a sixth aspect of the invention, there is provided a mobile terminal characterized in that the filter is arranged to operate adaptively according to the block types of the frame in the environment of the block boundary.

According to a seventh aspect of the invention, there is provided a storage medium characterized in that the software program further comprises machine executable steps for filtering adaptively according to the block types of the frame in the environment of the block boundary.

Because blocking artefacts only occur at block boundaries, according to the invention, filtering is advantageously only applied to pixels at block boundaries and the vicinity thereof. Edges that are part of the image can reside anywhere in the image area. In order that only pixels containing blocking artefacts are selected for corrective filtering and that the quality of edges that are part of the image itself is not affected during filtering, the following assumptions are made:

Changes in pixel values associated with edges that are part of the image are generally larger than those associated with blocking artefacts, and those edges within the image, where the pixel value change is small, do not suffer considerably from the rounding of the pixel value difference caused by filtering.

Because the image to be coded is generally divided into blocks both vertically and horizontally, the image contains both vertical and horizontal block boundaries. With regard to vertical block boundaries, there are pixels to the right and left of the boundary, and with regard to horizontal block boundaries, there are pixels above and below the boundary. In general, the location of the pixels can be described as being on a first or a second side of the block boundary. In an exemplary embodiment of the filtering method according to the invention, the number of pixels to be corrected, the characteristic features of the filter being used and the size of the filtering window depend on the following factors:

5

- a) The type of block on either side of the boundary (e.g. inter, copy, coded, not-coded),
- b) the difference in pixel values Δ across the block boundary. The difference can be defined in many ways. One definition is $\Delta = |r_1 - l_1|$, where r_1 is the value of the pixel on the first side of the block boundary closest to the boundary, and l_1 is the value of the pixel on the second side of the block boundary closest to the boundary,
- c) the size of the quantization step OP of the coefficients received as the result of the transformation used in coding, and
- d) differences in pixel values between the pixels on the first side of the block boundary, and correspondingly between the pixels on the second side of the block boundary.

In an advantageous embodiment of the method according to the invention, the number of the pixels selected for filtering can vary, and it is not necessarily the same on different sides of a block boundary. The number of pixels also depends on the type of block on either side of the boundary. Because the number of pixels is adapted to the general features of the image information contained by the frame in a particular region according to the factors mentioned above, the method produces a better filtering result than that provided by prior art methods. A "better" result in this context is one in which blocking artefacts are reduced to a greater extent while real edges in the image are affected to a lesser degree. This means that a larger amount of blocking artefacts can be removed without weakening the real image edges unreasonably.

It should be noted that in other embodiments of the invention, the factors affecting the filtering performed at a block boundary may differ from those presented above.

BRIEF DESCRIPTION OF THE DRAWINGS

In the following, the invention will be described in more detail with reference to the preferred embodiments and the accompanying drawings, in which

FIG. 1 represents a digital video encoding and decoding system (codec) according to prior art,

FIG. 2 represents the location of pixels in relation to a block boundary in an exemplary embodiment of the method according to the invention,

FIG. 3 represents alternatives for locating the filtering method according to the invention in a digital video encoding and decoding system,

FIG. 4 is a schematic representation of a device for implementing a method according to the invention,

FIG. 5 represents a device according to FIG. 4 in operation, and

FIG. 6 is a schematic representation of a portable video telecommunications device implementing a method according to the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT(S)

In the above, in connection with the description of the prior art, reference was made to FIG. 1. In the following description of the invention and its preferred embodiments, reference will be made mostly to FIGS. 2 to 5. The same reference numbers are used for corresponding parts in the figures.

FIG. 2 shows the location of the pixels r_1 - r_6 and l_1 - l_6 in relation to a vertical block boundary 30. To implement the method according to the invention, certain parameters are

6

specified. Parameter n is the largest number of pixels to be examined from the block boundary in one direction, and in the case of FIG. 2 its value is 6. It is advantageous to select the value of the parameter n so that it has a certain relation to both the difference of the pixel values Δ across the block boundary and to the size of the quantization step QP of the coefficients received as the result of image coding. Furthermore, the value of parameter n is advantageously smaller than or equal to the number of pixels in the block in the direction of examination, to avoid possible blocking artefacts associated with previous block boundaries spreading to the block boundary under examination. The following definition is recommended for use in a preferred embodiment of the invention applied to image blocks comprising 8x8 pixels:

$$n = \begin{cases} 0 & \Delta \geq 2.00\alpha \\ 1 & 1.50\alpha \leq \Delta < 2.00\alpha \\ 2 & 1.00\alpha \leq \Delta < 1.50\alpha \\ 3 & 0.66\alpha \leq \Delta < 1.00\alpha \\ 4 & 0.40\alpha \leq \Delta < 0.66\alpha \\ 5 & 0.25\alpha \leq \Delta < 0.40\alpha \\ 6 & 0 \leq \Delta < 0.25\alpha \end{cases} \quad (2)$$

wherein $\alpha = \text{QP} \cdot \log(\text{QP})$. If QP has a different value in blocks on different sides of the block boundary, the smaller value of QP is used in the calculations, as well as in all cases presented hereinafter, in which a definition includes reference to one OP value only. The invention does not place any limitations on the determination of the value of parameter n, but according to the guidelines of equation (2), it is advantageous that its value is generally higher when the difference of pixel values Δ across the block boundary is small in comparison with the size of the quantization step OP of the coefficients received as the result of the coding transformation. If the difference between the pixel values Δ is large, there is a high probability that there is a real image edge at the block boundary, and in this case the pixels are preferably not examined for filtering at all ($n=0$).

In the next step of the filtering method according to the invention, region type information concerning the two neighbouring blocks is examined, i.e. the type of the blocks on both sides of the block boundary in question is considered. According to the region type information, the value of the parameter n may further be limited (truncated) to provide even better results for reducing blocking artefacts. The region type information is included e.g. in the coded information relating to the pixels of a particular block, wherein that information is maintained, or temporarily stored, during decoding of the block until a truncated value n_{tr} for parameter n is determined.

Table 1 shows truncation values according to an advantageous embodiment of the present invention. Table 1 applies in a situation where the maximum value of n is 6, and, of course, different truncated values would be appropriate in situations where the maximum value is other than 6. The truncation values are used for the first and second sides of the block boundary depending on the region type of the block on the first side of the block boundary and on the region type of the block on the second side of the block boundary.

TABLE 1

type of the Block on the First side of a boundary	type of the Block on the Second side of a boundary							
	INTRA		COPY		CODED		NOT_CODED	
INTRA	n	n	2	2	n	4	n	2
COPY	2	2	2	2	2	4	2	2
CODED	4	n	4	2	4	4	4	2
NOT_CODED	2	n	2	2	2	4	2	2

Each cell of Table 1, corresponding to a particular region type combination, is split into two parts. The value on the left gives the truncation value $trval$ for the first side of the block boundary and the value on the right gives the truncation value $trval$ for the second side of the boundary. If the value of parameter n exceeds the value given in Table 1, n is truncated to the truncation value $trval$ in Table 1. If, however, the value of the overall activity parameter n does not exceed the value given in Table 1, the value of the parameter n (originally determined from equation (2)) is retained. In Table 1 the symbol “n” indicates that further truncation is not performed and the parameter value is retained. The truncated value n_{tr} for the parameter n can also be represented by the formula:

$$n_{tr} = \min(trval, n), \quad (3)$$

The same table can be used both for filtering across vertical block boundaries (horizontal filtering) by putting “Left”/“Right” in place of “First”/“Second” and for filtering across horizontal block boundaries (vertical filtering) by putting “Bottom”/“Up” in place of “First”/“Second”, respectively. Now, the value on the left gives the truncation value for the pixels on the left/below the block boundary, and the value on the right gives the truncation value for the pixels on the right/above the boundary.

To further clarify the use of Table 1, an example situation is presented in the following. In this illustrative example situation “horizontal filtering” is performed across a vertical block boundary 30. Assuming that the value for parameter n , calculated from equation 2, is e.g. 4, the block on the left-hand side of the block boundary 30 in question is of the Intra type, and the block on the right-hand side of the block boundary 30 in question is of the Not-coded type, Table 1 indicates that the truncation value for the left-hand side is n and the truncation value for the right-hand side is 2. This means that the 4 pixels closest to the block boundary (=the calculated value of n) are selected for examination from the left-hand side of the boundary, and 2 pixels closest to the block boundary (=the truncated value of n) are selected for examination from the right-hand side.

Another example situation is presented in the following. In this illustrative example situation “horizontal filtering” is performed across a vertical block boundary 30. Assuming that the value of parameter n calculated from equation (2) is e.g. 4, and the blocks on both sides of the block boundary 30 in question are of the Copy type, Table 1 indicates that the truncation value for the left-hand side is 2 and the truncation value for the right-hand side is 2. This means that the 2 pixels closest to the block boundary (=the truncated value of n) are selected for examination from the left-hand side of the boundary, and the 2 pixels closest to the block boundary (=the truncated value of n) are selected for examination from the right-hand side.

For bi-directionally predicted frames (B-frames), truncation of the parameter n is advantageously not applied because there is no unique block type information.

The next step in the filtering method according to the invention is to determine the values of the parameters d_r and d_l , which represent activity, or the differences of pixel values between pixels on one side of the block boundary. A preferred definition for d_r is the following:

$$d_r = 6, \text{ if } |r_1 - r_j| \leq \beta/j \text{ with all } j \in [1, 6],$$

otherwise: $d_r = i$, where i fulfils the conditions

$$i \in [1, n_{tr}],$$

$$|r_1 - r_{i+1}| > \beta/i \text{ and}$$

$$|r_1 - r_j| \leq \beta/j \text{ with all } j \in [1, i]. \quad (4)$$

Here, the auxiliary parameter $\beta = 4 \cdot \log(QP)$. The value of parameter d_l is determined similarly, except that all r 's are replaced by l 's and the corresponding truncated value n_{tr} for the parameter n must be used. The number 6 appears in definition (4) because, according to equation (2), the highest possible value of n is 6 in this case. If n is defined differently, but the parameters d_r and d_l are defined according to definition (4), the number 6 must be replaced by the highest possible value of n according to the new definition.

It is advantageous that the values of the parameters d_r and d_l are calculated independent of one another, because the image information contained by the frame can be different on different sides of the block boundary. The invention does not limit the definition of parameters d_r and d_l , but according to the guidelines of definition (4), it is advantageous that these parameters are used to limit the blocking artefact processing relatively close to the block boundary, if there is a real image edge near the block boundary. The essential features of definition (4) can be summarised as follows: the value of parameter d_r (and correspondingly the value of parameter d_l) provides an indication of how many pixels counted from the block boundary have approximately the same value as the pixel at the block boundary.

A high value of parameter n (e.g. 6) indicates that the difference between the pixel values at the block boundary is relatively small compared with the general variation of the pixel values within the block. In this case, it is possible that there is a real image edge near the block boundary. By selecting a sufficiently small value of parameter d_r (or d_l), it is possible to restrict the filtering aimed at correcting blocking artefacts so that it does not have a deteriorating effect on a real image edge close to the block boundary. In some situations, a large number of pixels counted from the block boundary have approximately the same value as the pixel at the block boundary. In that case, definition (4) would give the parameter d_r (or d_l) a relatively high value. However, if there is a clear discontinuity in pixel values between the blocks, the parameter n has a small value and the truncated value n_{tr} is used in the definition (4) which make sure that an unreasonably high value is not selected as the value of the parameter d_r (or d_l). Otherwise, a relatively high value of the parameter d_r (or d_l) would result in unnecessary filtering.

If blocks on both sides of the block boundary are Intra-type blocks, the truncation has no effect on the selection of the parameter values n , d_r and d_l . On the other hand, if at least one of the blocks has a type other than Intra, the truncation of the value n according to the formula (3) may limit the number of pixels filtered. This has the advantage that the block boundaries are not smoothed too much.

In addition, the largest possible number of pixels to be filtered must be decided. This does not have a notation of its own in FIG. 2, but it can be e.g. 3, which means that filtering can only be used to correct the value of the pixels r_1 , r_2 , r_3 , l_1 , l_2 and l_3 .

When the values of the parameters n , n_r , d_r and d_l have been determined, filtering is carried out using a suitable filter. The invention does not limit the kind of filter that can be used, but a filtering arrangement that has been found preferable, will be described in the following. Filtering is used to determine a new value for the pixels selected for filtering. In a preferred embodiment of the invention, a new pixel value is determined for a given pixel by calculating the mean of the pixel values that appear in a filtering window. In the preferred embodiment, the filtering window is symmetrical with regard to the pixel to be filtered and contains, in addition to the pixel to be filtered, one, two or three pixels from both sides, depending on the values of the parameters d_r and d_l as described hereinafter. Of course, these are only examples and other values could be chosen in situations where n , n_r , d_r and d_l are defined differently. The calculated mean value is rounded to the closest digitized pixel value, whereby it becomes the new value of the filtered pixel.

Table 2 shows the determination of the width of the filtering window for the pixels r_1 , r_2 and r_3 according to the value of parameter d_r in a preferred embodiment of the invention. The values of the pixels l_1 , l_2 and l_3 are determined in the same manner according to the value of the parameter d_l . In the table, "X" means that the pixel in question is not filtered at all, and the number means that the filtering window includes a number of pixels shown by the number from each side of the pixel being examined. Among other things, Table 2 shows that for filtering to be applied to any pixel, parameters d_r and d_l must both have a value greater than 1.

d_r ($d_l > 1$)	r_1	r_2	r_3
1	X	X	X
2	1	X	X
3	1	1*	X
4	2	2	X
5	2	2	2**
6	3 or 2***	3	3

*the filtered value of pixel r_1 is used for filtering of pixel r_2

**the filtered values of pixels r_1 and r_2 are used for filtering pixel r_3

***3 if $d_r > 2$, otherwise 2.

The above description relates to implementing the filtering on one horizontal part of a pixel row, which is 12 pixels long and located symmetrically on both sides of a vertical block boundary. The description can be easily generalized to concern vertical parts of pixel columns, which are located symmetrically on both sides of a horizontal block boundary: FIG. 2 can be turned 90 degrees counter-clockwise, whereby block boundary 30 becomes horizontal, and the pixels shown in the figure form part of the vertical pixel column so that pixels r_1 - r_6 are the pixels above and pixels l_1 - l_6 are the pixels below the boundary. To filter block boundaries throughout the whole frame, applying the method according to the invention, all vertical block boundaries of the frame are examined row by row and all horizontal block boundaries are examined column by column. The order has no significance as such, and thus all the horizontal block boundaries of the frame could be examined first column by column, and then all the vertical block boundaries row by row. In an advantageous embodiment of the invention, filtering is repeated line by line, i.e. the first line of the pixels in the blocks (besides the boundary) is filtered first, then the second line, etc.

FIG. 3 shows at which points a digital image encoding and decoding system (codec) according to prior art can be

improved by applying filtering according to the invention. The first alternative is to place a block implementing the filtering method according to the invention in the output of the decoder of the receiver, as illustrated by reference number 31. In this case, block boundaries in the video frame are filtered after all blocks within the frame have been decoded. This requires block type information to be stored for all the blocks in one frame. Another alternative is to place the filtering block in the receiver before the point at which the decoded frame is directed to the frame memory 25 for forming a prediction frame, as illustrated by reference number 32. In this case, block type information for all blocks within the frame must also be stored, as block boundary filtering is still performed after decoding and reconstructing the entire frame. However, this alternative has the advantage that the removal of blocking artefacts also has an effect on the formation of a prediction frame, whereby blocking artefacts in one frame are not propagated via the prediction frame to subsequent frames. In order to achieve the effect just described, the block that performs filtering according to the invention can be placed either before or after the frame memory 25. However, the location shown by reference number 32 is preferred, because when applied at this stage, the filtering influences simultaneously the frame to be output by the receiving decoder and the frame to be saved in the memory. In the transmitter, a block implementing the filtering method according to the invention can be placed as shown by reference numbers 33 and 34, either before or after the frame memory 16. In this way, the invention can also be used to produce a corrected prediction frame at the transmission end.

In a particularly advantageous embodiment of the invention, the block carrying out the filtering according to the invention is implemented in a digital signal processor or a corresponding device suited for processing a digital signal, which can be programmed to apply predetermined processing functions to a signal received as input data. FIG. 4 presents the functional elements of a block carrying out the filtering method according to the invention, implemented as a digital signal processor. Correspondingly, operation of the filtering block is illustrated by FIG. 5. Referring first to FIG. 4, the functional elements are programmed with functional definitions (35-39) for calculating the parameters that control the filtering method according to the invention. Advantageously, the functional definitions are loaded into the signal processor in connection with its manufacture or programming. During operation according to FIG. 5, the frame is saved temporarily in register 41, so that it can be processed by the signal processor. Processing of the frame proceeds block-by-block. At a given instant, a number of pixels indicated by the parameter n , n_r , according to the definitions provided in Table 1, are selected to be examined 42 from each side of a certain point on a certain block boundary, the d -parameters 43 are calculated, and filtering 44 is performed. These operations are repeated until all boundaries of all blocks have been filtered/processed, after which the frame can be output from register 41 and a new frame saved for processing. The measures according to FIGS. 4 and 5 can be carried out in a separate signal processor or they can be implemented in a general processor which also contains other arrangements for signal processing.

A storage medium can be used for storing a software program comprising machine executable steps for performing the method according to the invention. In an advantageous embodiment of the invention, the software program is

11

read from the storage medium to a device comprising programmable means, e.g. a processor, for performing the method of the invention.

The invention can be modified without departing from the scope defined by the claims presented hereinafter, using the capabilities of a person skilled in the art without actual inventive steps. For example, the parameter Δ can be calculated using the formula $\Delta = |r_1 + r_2| - (l_1 + l_2)$ or some other formula considered suitable. The definitions of the other parameters presented above should also be considered as examples only. A particularly advantageous use of the invention is in mobile video telecommunication applications, digital television receivers and other devices that at least receive and decode digital video image. It should further be noted that in general the method according to the invention may be applied to any image coding method in which an image is encoded/decoded on a block-by-block basis, including individual (i.e. still) digital images.

FIG. 6 shows a mobile terminal 46 intended for use as a portable video telecommunications device and applying the method according to the invention. Advantageously, the mobile terminal 46 comprises at least display means 47 for displaying images, audio means 48 for capturing and reproducing audio information, a keyboard 49 for inputting e.g. user commands, a radio part 50 for communicating with a mobile communications network (not shown), processing means 51 for controlling the operation of the device, memory means 52 for storing information, and preferably a camera 53 for taking images.

The present invention is not solely restricted to the above presented embodiments, but it can be modified within the scope of the appended claims.

What is claimed is:

1. A method, comprising:

decoding a first encoded image block, which has been encoded using a first type of prediction encoding method, and a second encoded image block, which has been encoded using a second type of prediction encoding method, in accordance with their respective prediction encoding methods to form a first decoded image block and a second decoded image block, the first type of prediction encoding method and the second type of prediction encoding method being selected from a group of prediction encoding methods comprising at least: intra coding, copy coding, motion-compensated prediction coding, and not-coded coding, a block boundary being formed between the first and second decoded image blocks, the first decoded image block being on a first side of the block boundary, the second decoded image block being on a second side of the block boundary;

providing information on the first and second prediction encoding methods to a block boundary filter; and

performing, by the block boundary filter, an adaptive block boundary filtering operation on the block boundary formed between the first decoded image block on the first side of the block boundary and the second decoded image block on the second side of the block boundary, the method further comprising:

determining, by the block boundary filter, a first number of pixels to be examined on the first side of the block boundary and a second number of pixels to be examined on the second side of the block boundary as a parameter of the adaptive block boundary filtering operation, based on the types of the first and the second prediction encoding methods.

12

2. A method according to claim 1, wherein the adaptive block boundary filtering operation performed on the block boundary by the block boundary filter is dependent at least in part on a region type of an image block on a first side of the block boundary and a region type of an image block on a second side of the block boundary.

3. A method according to claim 1, wherein the number of pixels selected for examination depends on a difference in pixel value between pixels across the block boundary.

4. A method according to claim 1, wherein the number of pixels selected for examination depends on the size of a quantization step used to quantize coefficients used in encoding the image blocks.

5. A method according to claim 1, comprising selecting certain pixels to be filtered and determining a new value for each pixel to be filtered on the basis of pixels that appear in a filtering window set around the pixel.

6. A method according to claim 5, wherein the new value of the pixel is the mean value of the pixels that appear in the filtering window.

7. A method according to claim 1, comprising selecting pixels to be filtered from the pixels selected for examination.

8. A method according to claim 1, wherein the first and second prediction encoding methods are of the same type.

9. A digital signal processor configured to perform a method for reducing visual artefacts due to block boundaries according to claim 1.

10. A block boundary filter configured to perform an adaptive block boundary filtering operation on a block boundary formed between a first decoded image block on a first side of the block boundary and a second decoded image block on a second side of the block boundary, the first decoded image block having been encoded using a first type of prediction encoding method and the second decoded image block having been encoded using a second type of prediction encoding method, the first type of prediction encoding method and the second type of prediction encoding method being selected from a group of prediction encoding methods comprising at least: intra coding, copy coding, motion-compensated prediction coding, and not-coded coding, the block boundary filter being configured to receive information on the types of the first and second prediction encoding methods, and to determine a first number of pixels to be examined on the first side of the block boundary and a second number of pixels to be examined on the second side of the block boundary as a parameter of the adaptive block boundary filtering operation, based on the types of the first and the second prediction encoding methods.

11. A block boundary filter according to claim 10, configured to perform said adaptive block boundary filtering operation on the block boundary in dependence at least in part on a region type of an image block on a first side of the block boundary and a region type of an image block on a second side of the block boundary.

12. A block boundary filter according to claim 10, further configured to select said number of pixels for examination in dependence on the difference in pixel value between pixels across the block boundary.

13. A block boundary filter according to claim 10, wherein the filter is configured to select a number of pixels for examination in dependence on the size of a quantization step used to quantize coefficients used in encoding the image blocks.

14. A block boundary filter according to claim 10, wherein the filter is configured to select certain pixels to be filtered

13

and to determine a new value for each pixel to be filtered on the basis of pixels that appear in a filtering window set around the pixel.

15 15. A block boundary filter according to claim 14, wherein the filter is configured to calculate the new value for each pixel to be filtered as a mean value of the pixels that appear in the filtering window.

16. A block boundary filter according to claim 10, wherein the first and second prediction encoding methods are of the same type.

17. A video encoder comprising a coding block, a decoding block and a prediction block for prediction encoding a frame of a digital video signal by blocks, the video encoder comprising a block boundary filter for reducing visual artifacts due to a block boundary between a first decoded image block on a first side of the block boundary and a second decoded image block on a second side of the block boundary, the first decoded image block having been encoded using a first type of prediction encoding method and the second decoded image block having been encoded using a second type of prediction encoding method, the first type of prediction encoding method and the second type of prediction encoding method being selected from a group of prediction encoding methods comprising at least: intra coding, copy coding, motion-compensated prediction coding, and not-coded coding, wherein the block boundary filter is configured to receive information on the selected type of the first prediction encoding method and the selected type of the second prediction encoding method and to operate adaptively according to the block types of the frame in the environment of the block boundary to determine a first number of pixels to be examined on the first side of the block boundary and a second number of pixels to be examined on the second side of the block boundary, as a parameter of the filtering operation, based on the selected types of the first and second prediction encoding method.

18. A video decoder comprising a decoding block and a prediction block for decoding a frame of a digital video signal by blocks, the video decoder comprising a block boundary filter for reducing visual artifacts due to a block boundary between a first decoded image block on a first side of the block boundary and a second decoded image block on a second side of the block boundary, the first decoded image block having been encoded using a first type of prediction encoding method and the second decoded image block having been encoded using a second type of prediction encoding method, the first type of prediction encoding method and the second type of prediction encoding method being selected from a group of prediction encoding methods comprising at least: intra coding, copy coding, motion-compensated prediction coding, and not-coded coding, wherein the block boundary filter is configured to receive information on the selected type of the first prediction encoding method and the selected type of the second prediction encoding method, and to operate adaptively according to the block types of the frame in the environment of the block boundary to determine a first number of pixels to be examined on the first side of the block boundary and a second number of pixels to be examined on the second side of the block boundary, as a parameter of the filtering operation, based on the selected types of the first and second prediction encoding method.

19. A video codec comprising a coding block, a decoding block and a prediction block for prediction encoding a frame of a digital video signal by blocks, a block type being defined according to the prediction encoding method for a block selected according to a predetermined set of coding

14

types, the video codec comprising a block boundary filter for reducing visual artifacts due to a block boundary between a first decoded image block on a first side of the block boundary and a second decoded image block on a second side of the block boundary, the first decoded image block having been encoded using a first type of prediction encoding method and the second decoded image block having been encoded using a second type of prediction encoding method, the first type of prediction encoding method and the second type of prediction encoding method being selected from a group of prediction encoding methods comprising at least: intra coding, copy coding, motion-compensated prediction coding, and not-coded coding, wherein the block boundary filter is configured to receive information on the selected type of the first prediction encoding method and the selected type of the second prediction encoding method, and to operate adaptively according to the block types of the frame in the environment of the block boundary to determine a first number of pixels to be examined on the first side of the block boundary and a second number of pixels to be examined on the second side of the block boundary, as a parameter of the filtering operation, based on the selected types of the first and second prediction encoding method.

20. A mobile terminal comprising a video codec, the video codec comprising a coding block, a first decoding block and a first prediction block for prediction encoding a frame of a digital video signal by blocks, and a second decoding block and a second prediction block for decoding a prediction encoded frame of the digital video signal by blocks, a block type being defined according to the prediction encoding method for a block selected according to a predetermined set of coding types, the video codec comprising a block boundary filter for reducing visual artifacts due to a block boundary between a first decoded image block on a first side of the block boundary and a second decoded image block on a second side of the block boundary, the first decoded image block having been encoded using a first type of prediction encoding method and the second decoded image block having been encoded using a second type of prediction encoding method, the first type of prediction encoding method and the second type of prediction encoding method being selected from a group of prediction encoding methods comprising at least: intra coding, copy coding, motion-compensated prediction coding, and not-coded coding, wherein the block boundary filter is configured to receive information on the selected type of the first prediction encoding method and the selected type of the second prediction encoding method, and to operate adaptively according to the block types of the frame in the environment of the block boundary to determine a first number of pixels to be examined on the first side of the block boundary and a second number of pixels to be examined on the second side of the block boundary, as a parameter of the filtering operation, based on the selected types of the first and second prediction encoding method.

21. A storage medium comprising a software program for reducing visual artifacts due to block boundaries between decoded image blocks in a frame of a digital video signal, the software program comprising machine executable code for performing an adaptive block boundary filtering operation on a block boundary formed between a first decoded image block on a first side of the block boundary and a second decoded image block on a second side of the block boundary, the first decoded image block having been encoded using a first type of prediction encoding method and the second decoded image block having been encoded using a second type of prediction encoding method, the first

15

type of prediction encoding method and the second type of prediction encoding method being selected from a group of prediction encoding methods comprising at least: intra coding, copy coding, motion-compensated prediction coding, and not-coded coding, wherein the software program comprises machine executable code for receiving information on the selected type of the first prediction encoding method and the selected type of the second prediction encoding method and machine executable code for determining a first number of pixels to be filtered on the first side of the block boundary and a second number of pixels to be examined on the second side of the block boundary, as a parameter of the adaptive block boundary filtering operation, based on the selected types of the first and second prediction encoding method.

22. A storage medium according to claim 21, wherein the first and second prediction encoding methods are of the same type.

23. A method of video encoding comprising:

performing an adaptive block boundary filtering operation on a block boundary formed between a first decoded image block on a first side of the block boundary and a second decoded image block on a second side of the block boundary using an adaptive block boundary filter, the first decoded image block having been encoded using a first type of prediction encoding method and the second decoded image block having been encoded using a second type of prediction encoding method, the first type of prediction encoding method and the second type of prediction encoding method being selected from a group of prediction encoding methods comprising at least: intra coding, copy coding, motion-compensated prediction coding, and not-coded coding, the adaptive block boundary filtering operation comprising:

examining, by the adaptive block boundary filter, the type of the first prediction encoding method and the type of the second prediction encoding method; and

determining, by the adaptive block boundary filter, a first number of pixels to be examined on the first side of the block boundary and a second number of pixels to be examined on the second side of the block boundary, as a parameter of the filtering operation, based on the types of the first and second prediction encoding methods.

24. A method of video encoding according to claim 23, wherein the first and second prediction encoding methods are of the same type.

25. A method of video decoding, comprising:

performing an adaptive block boundary filtering operation on a block boundary formed between a first decoded image block on a first side of the block boundary and a second decoded image block on a second side of the block boundary using an adaptive block boundary filter, the first decoded image block having been encoded using a first type of prediction encoding method and the second decoded image block having been encoded using a second type of prediction encoding method, the first type of prediction encoding method and the second type of prediction encoding method being selected from a group of prediction encoding methods comprising at least: intra coding, copy coding, motion-compensated prediction coding, and not-coded coding, the block boundary filtering operation comprising:

examining, by the adaptive block boundary filter, the type of the first prediction encoding method and the type of the second prediction encoding method; and

16

determining, by the adaptive block boundary filter, a first number of pixels to be examined on the first side of the block boundary and a second number of pixels to be examined on the second side of the block boundary as a parameter of the filtering operation based on the types of the first and second prediction encoding method.

26. A method of video decoding according to claim 25, wherein the first and second prediction encoding methods are of the same type.

27. A video encoder comprising an adaptive block boundary filter configured to perform an adaptive block boundary filtering operation on a block boundary formed between a first decoded image block on a first side of the block boundary and a second decoded image block on a second side of the block boundary, the first decoded image block having been encoded using a first type of prediction encoding method and the second decoded image block having been encoded using a second type of prediction encoding method, the first type of prediction encoding method and the second type of prediction encoding method being selected from a group of prediction encoding methods comprising at least: intra coding, copy coding, motion-compensated prediction coding, and not-coded coding, wherein the adaptive block boundary filter is configured to examine the type of the first prediction encoding method and the type of the second prediction encoding method, and to determine a first number of pixels to be examined on the first side of the block boundary and a second number of pixels to be examined on the second side of the block boundary, as a parameter of the adaptive block boundary filtering operation, based on the types of the first and second prediction encoding method.

28. A video encoder according to claim 27, wherein the first and second prediction encoding methods are of the same type.

29. A video decoder comprising an adaptive block boundary filter configured to perform an adaptive block boundary filtering operation on a block boundary formed between a first decoded image block on a first side of the block boundary and a second decoded image block on a second side of the block boundary, the first decoded image block having been encoded using a first type of prediction encoding method and the second decoded image block having been encoded using a second type of prediction encoding method, the first type of prediction encoding method and the second type of prediction encoding method being selected from a group of prediction encoding methods comprising at least: intra coding, copy coding, motion-compensated prediction coding, and not-coded coding, wherein the adaptive block boundary filter is configured to examine the type of the first prediction encoding method and the type of the second prediction encoding method, and to determine a first number of pixels to be examined on the first side of the block boundary and a second number of pixels to be examined on the second side of the block boundary, as a parameter of the adaptive block boundary filtering operation, based on the types of the first and second prediction encoding method.

30. A video decoder according to claim 29, wherein the first and second prediction encoding methods are of the same type.

31. A video codec comprising an adaptive block boundary filter configured to perform an adaptive block boundary filtering operation on a block boundary formed between a first decoded image block on a first side of the block boundary and a second decoded image block on a second side of the block boundary, the first decoded image block having been encoded using a first type of prediction encoding method and the second decoded image block having

17

been encoded using a second type of prediction encoding method, the first type of prediction encoding method and the second type of prediction encoding method being selected from a group of prediction encoding methods comprising at least: intra coding, copy coding, motion-compensated prediction coding, and not-coded coding, wherein the adaptive block boundary filter is configured to examine the type of the first prediction encoding method and the type of the second prediction encoding method, and to determine a first number of pixels to be examined on the first side of the block boundary and a second number of pixels to be examined on the second side of the block boundary, as a parameter of the adaptive block boundary filtering operation, based on the types of the first and second prediction encoding method.

32. A video codec according to claim 31, wherein the first and second prediction encoding methods are of the same type.

33. A mobile terminal comprising an adaptive block boundary filter configured to perform an adaptive block boundary filtering operation on a block boundary formed between a first decoded image block on a first side of the block boundary and a second decoded image block on a second side of the block boundary, the first decoded image block having been encoded using a first type of prediction encoding method and the second decoded image block having been encoded using a second type of prediction encoding method, the first type of prediction encoding method and the second type of prediction encoding method being selected from a group of prediction encoding methods comprising at least: intra coding, copy coding, motion-compensated prediction coding, and not-coded coding, wherein the adaptive block boundary filter is configured to examine the type of the first prediction encoding method and the type of the second prediction encoding method, and to determine a first number of pixels to be examined on the first side of the block boundary and a second number of pixels to be examined on the second side of the block boundary, as a parameter of the adaptive block boundary filtering operation, based on the types of the first and second prediction encoding method.

34. A mobile terminal according to claim 33, wherein the first and second prediction encoding methods are of the same type.

35. A digital signal processor comprising a filtering block configured to perform an adaptive block boundary filtering operation on a block boundary formed between a first decoded image block on a first side of the block boundary and a second decoded image block on a second side of the block boundary, the first decoded image block having been encoded using a first type of prediction encoding method and the second decoded image block having been encoded using a second type of prediction encoding method, the first type of prediction encoding method and the second type of prediction encoding method being selected from a group of prediction encoding methods comprising at least: intra coding, copy coding, motion-compensated prediction coding, and not-coded coding, wherein the filtering block is configured to examine the type of the first prediction encoding method and the second prediction encoding method, and to determine a first number of pixels to be examined on the first side of the block boundary and a second number of pixels to be examined on the second side of the block boundary, as a

18

parameter of the adaptive block boundary filtering operation, based on the types of the first and second prediction encoding method.

36. A method, comprising performing a filtering operation on a block boundary that is dependent at least in part on a first prediction encoding method used to encode first image block on a first side of the block boundary and a second prediction encoding method used to encode a second image block on a second side of the block boundary, wherein the first and the second prediction encoding methods are selected from a group of prediction encoding methods comprising at least: intra coding, copy coding, motion-compensated prediction coding, and not-coded coding.

37. A apparatus comprising a filter configured to examine a prediction encoding method used to encode a first image block on a first side of a block boundary and a prediction encoding method used to encode a second image block on a second side of the block boundary and to perform a filtering operation on the block boundary in dependence at least in part on the first prediction encoding method used to encode the first image block on the first side of the block boundary and the second prediction encoding method used to encode the second image block on the second side of the block boundary, wherein the first and the second prediction encoding methods are selected from a group of prediction encoding methods comprising at least: intra coding, copy coding, motion-compensated prediction coding, and not-coded coding.

38. A video encoder comprising a filter according to claim 37.

39. A video decoder comprising a filter according to claim 37.

40. A mobile terminal comprising a filter according to claim 37.

41. A method comprising:

using an adaptive block boundary filter to perform an adaptive block boundary filtering operation on a block boundary formed between a first decoded image block on a first side of the block boundary and a second decoded image block on a second side of the block boundary, the first decoded image block having been encoded using a first type of prediction encoding method and the second decoded image block having been encoded using a second type of prediction encoding method, the first type of prediction encoding method and the second type of prediction encoding method being selected from a group of prediction encoding methods comprising at least: intra coding, copy coding, motion-compensated prediction coding, and not-coded coding, the method further comprising: receiving, by the adaptive block boundary filter, information on the type of the first prediction encoding method and the type of the second prediction encoding method, and

determining, by the adaptive block boundary filter, a first number of pixels to be examined on the first side of the block boundary and a second number of pixels to be examined on the second side of the block boundary, as a parameter of the filtering operation, based at least in part upon the types of the first and second prediction encoding methods.

* * * * *

EXHIBIT 12



US006968005B2

(12) **United States Patent**
Hannuksela

(10) **Patent No.:** **US 6,968,005 B2**
(45) **Date of Patent:** **Nov. 22, 2005**

(54) **VIDEO CODING**

(75) Inventor: **Miska Hannuksela**, Tampere (FI)

(73) Assignee: **Nokia Mobile Phones Limited**, Espoo (FI)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 614 days.

(21) Appl. No.: **09/855,640**

(22) Filed: **May 15, 2001**

(65) **Prior Publication Data**

US 2002/0021752 A1 Feb. 21, 2002

(30) **Foreign Application Priority Data**

May 15, 2000 (GB) 0011597

(51) **Int. Cl.**⁷ **H04N 7/12**

(52) **U.S. Cl.** **375/240.01**

(58) **Field of Search** 375/240.01, 240.12, 375/240.13, 240.15, 240.16; 348/394.1, 401.1, 402.1, 412.1, 415.1; 382/236, 238; H04N 7/12

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,972,261	A	11/1990	Whalley	
5,768,527	A	6/1998	Zhu et al.	
6,111,915	A *	8/2000	Fukunaga et al. 375/240.12
6,169,821	B1 *	1/2001	Fukunaga et al. 382/239
6,357,028	B1 *	3/2002	Zhu 714/751
6,393,152	B2 *	5/2002	Takahashi et al. 382/233
6,629,261	B1 *	9/2003	Chintada et al. 714/4
6,711,140	B1 *	3/2004	Agarwal et al. 370/324

FOREIGN PATENT DOCUMENTS

CN	1199531	A	11/1998
EP	0637027	A2	2/1995
EP	0702492	A1	3/1996
EP	0713340	A2	5/1998

EP	0849952	6/1998
EP	0851685	7/1998
GB	2268661	1/1994

OTHER PUBLICATIONS

ITU—Telecommunications Standardization Sector, Ninth Meeting: Red Bank, NJ, Oct. 19–22, 1999 p. 1–2.

ITU—Telecommunications Standardization Sector, Seventh Meeting: Feb. 15–19, 1999, Monterey, Ca, USA pp. 1–11.

“Error Concealment in Encoded Video Streams” by Salma, et al pp. 1–32.

“Professor Delp’s Recent Publications” Recent Pub. of Edward J. Delp.

“Picture Layer” 5.1 Recommendation H.263 (Feb. 1998).

“Error Control and Concealment for Video Communication: A Review” pp. 974–997.

ITU—Telecommunications Standardization Sector, Ninth Meeting: Red Bank, New Jersey, Oct. 19–22, 1999 p. 1.

ITU—Telecommunications Standardization Sector, Ninth Meeting: Red Bank, New Jersey, Oct. 19–22, 1999 pp 1–3.

ITU—Telecommunications Standardization Sector, Ninth Meeting: Red Bank, New Jersey, Oct. 19–22, 1999.

ITU—Telecommunications Standardization Sector, Nov., 1999, pp. 1–43.

* cited by examiner

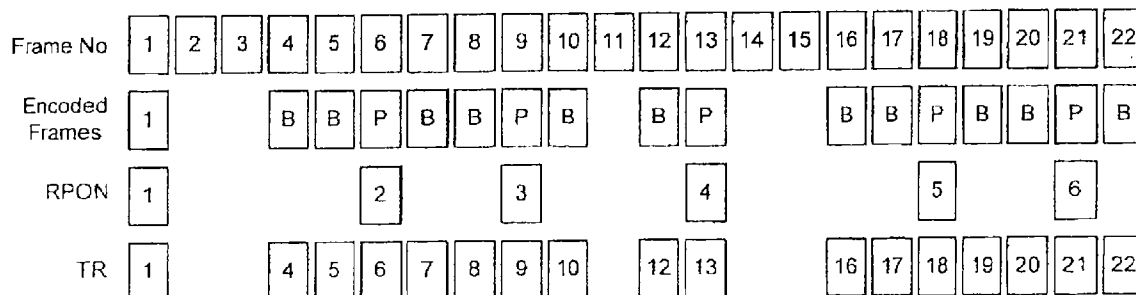
Primary Examiner—Nhon Diep

(74) *Attorney, Agent, or Firm*—Antonelli, Terry, Stout and Kraus, LLP.

(57) **ABSTRACT**

A method of encoding a video signal representing a sequence of pictures, the method employing both non-temporal prediction and temporal prediction, wherein the method comprises, for each picture that forms a reference picture for the temporal prediction of another picture, associating with each such picture an indicator indicating the temporal order of the reference picture in the encoded video signal relative to the other reference pictures in the encoded video signal.

46 Claims, 6 Drawing Sheets



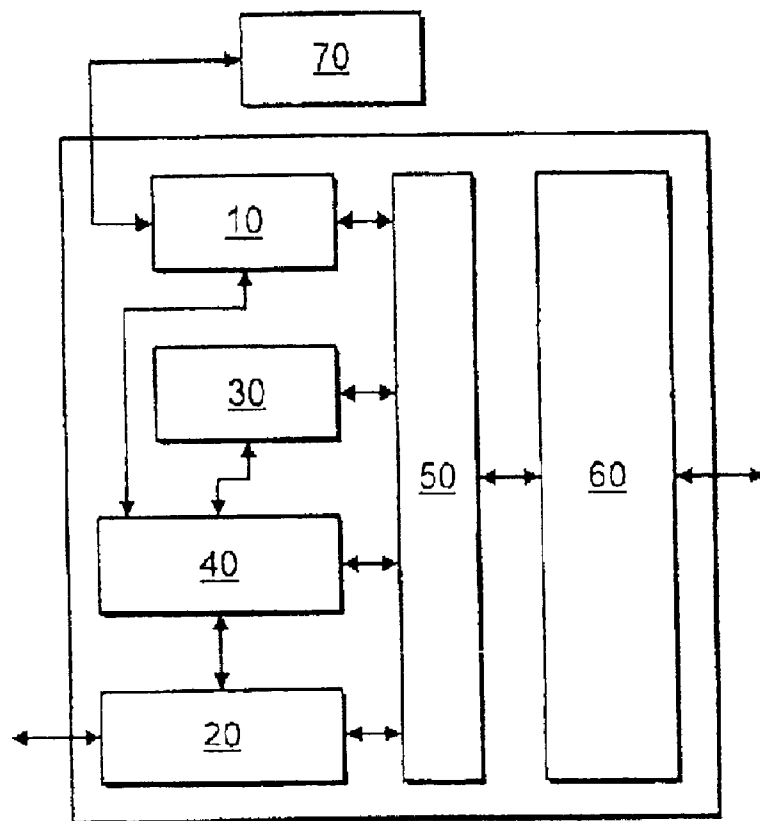
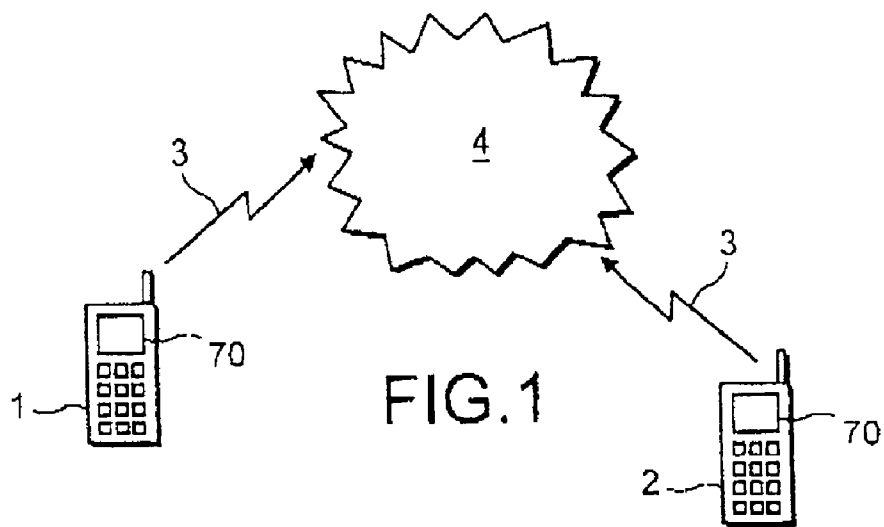


FIG. 2

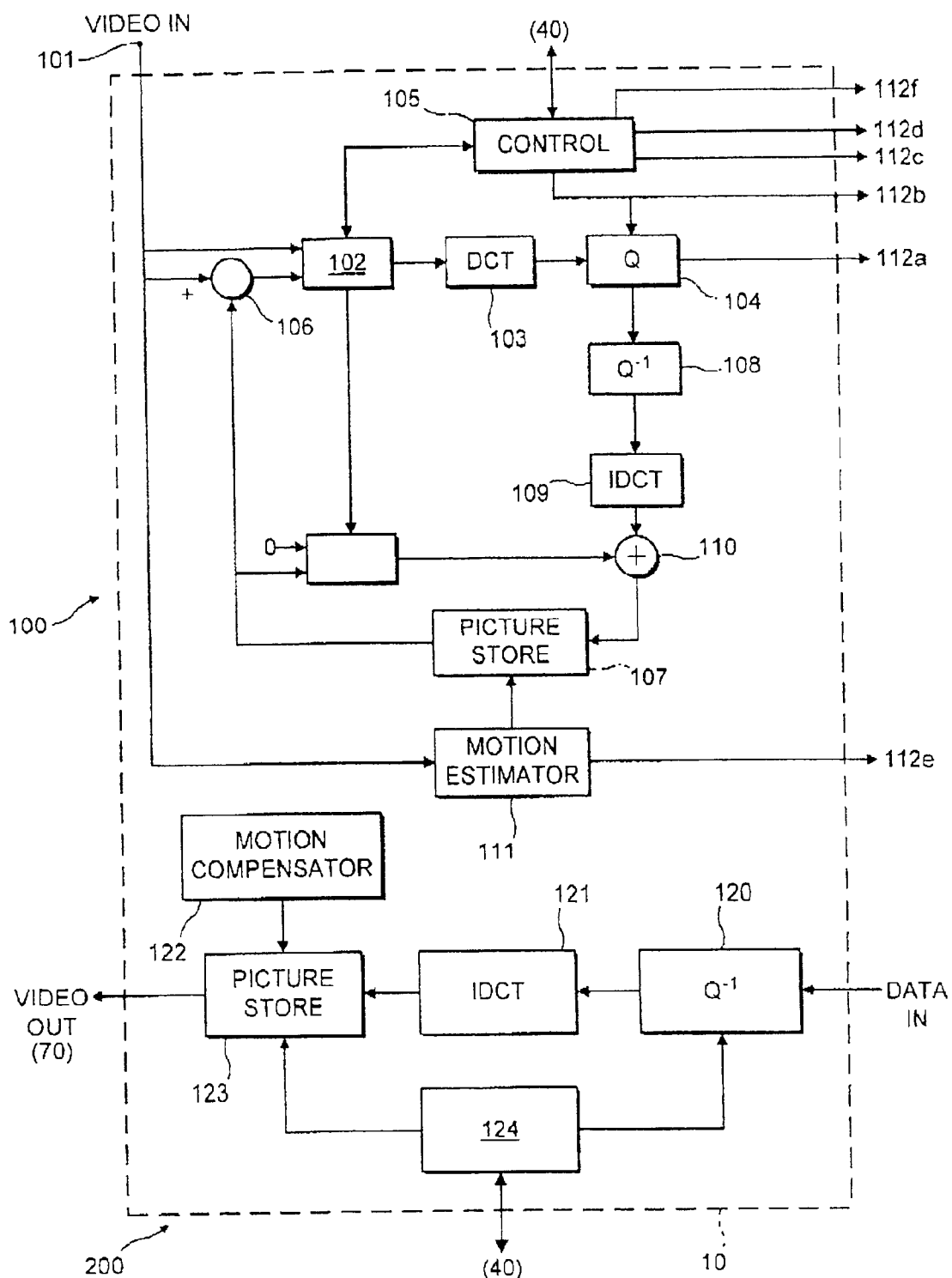


FIG. 3

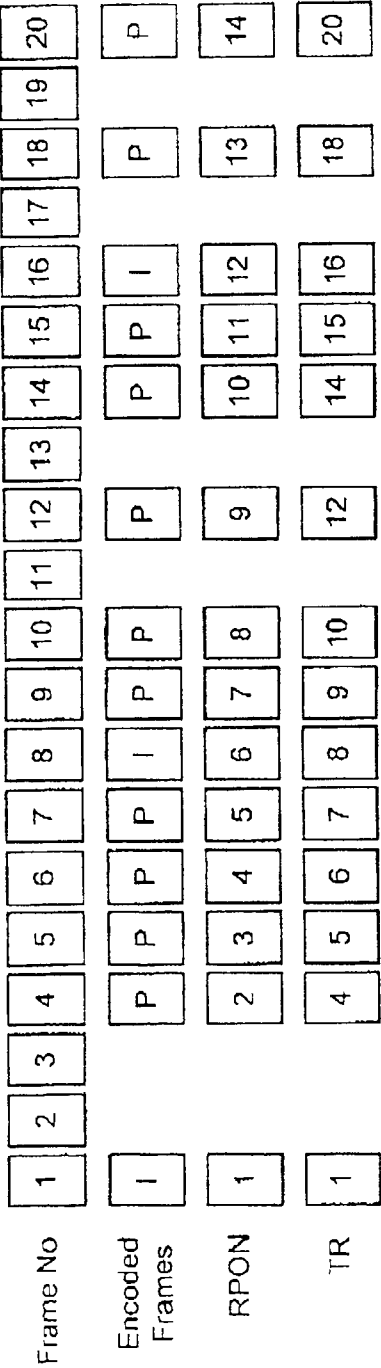


FIG. 4

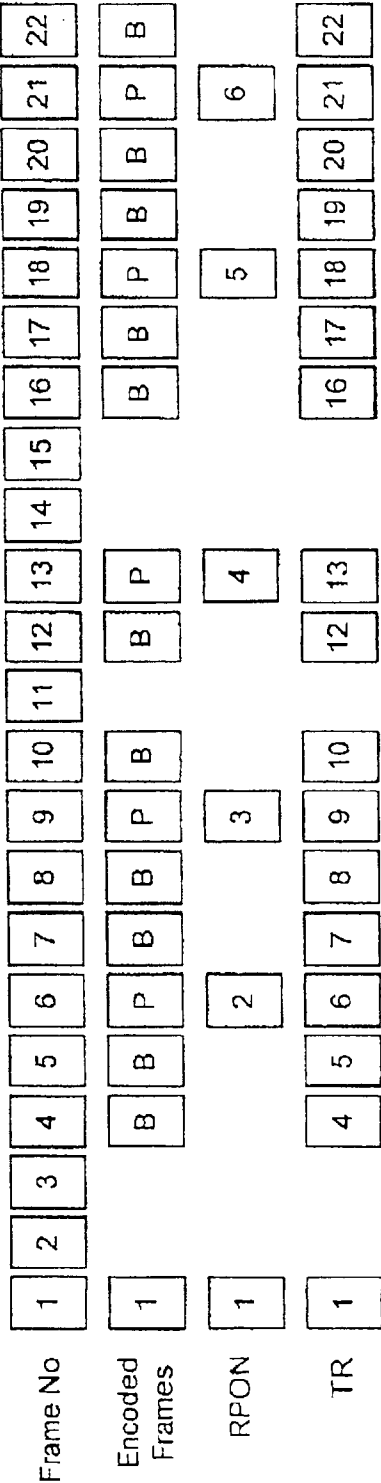


FIG. 5

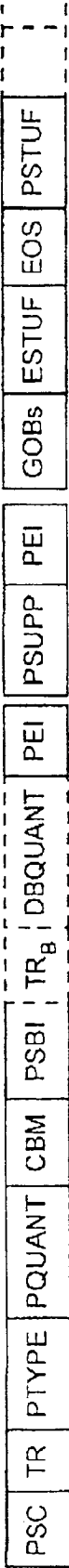


FIG. 6

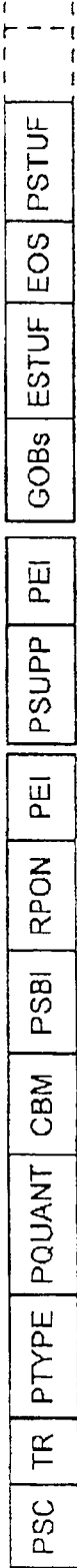


FIG. 7

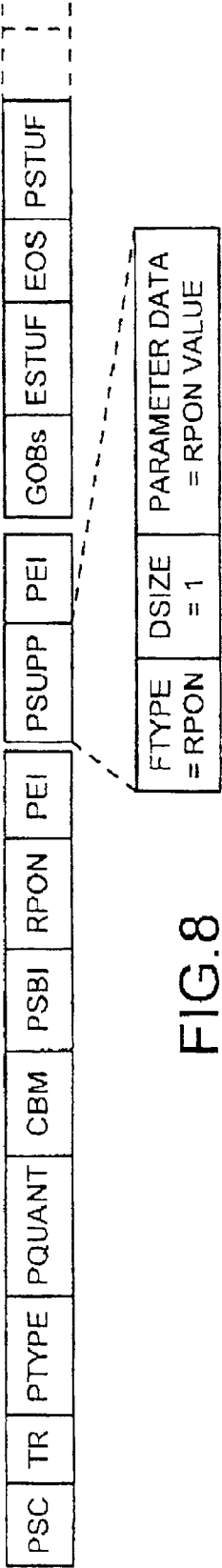


FIG. 8

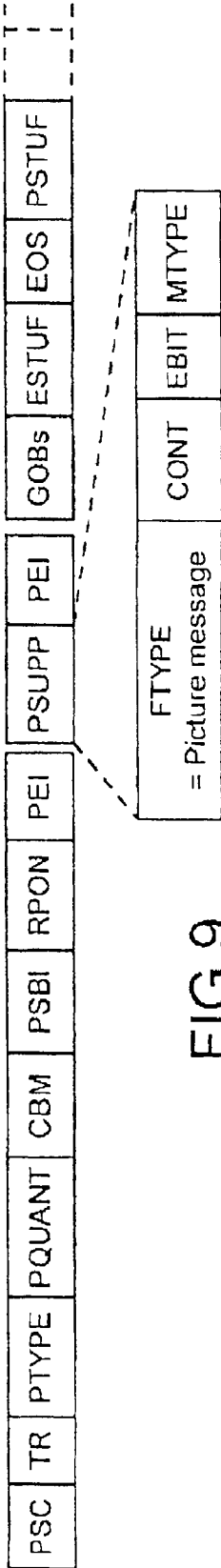
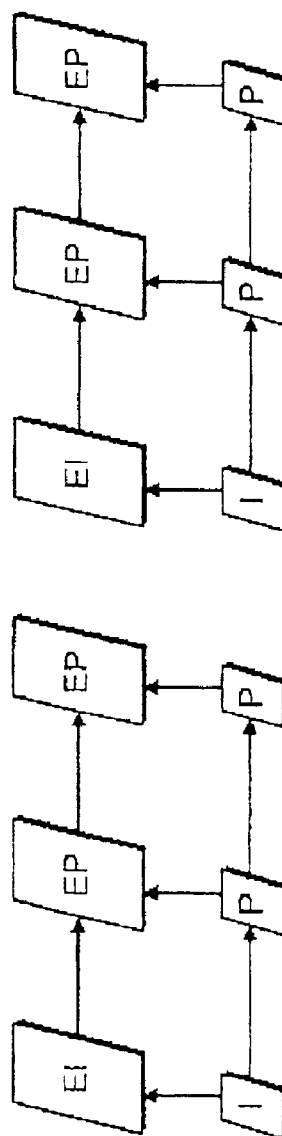
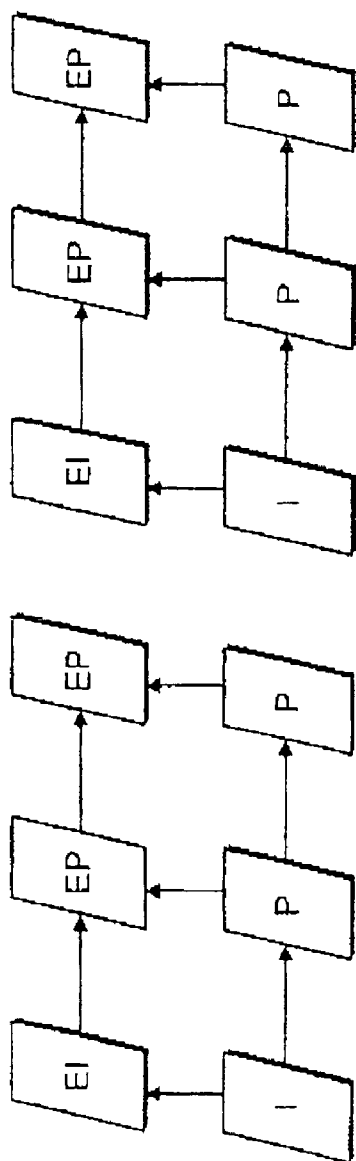


FIG. 9



1

VIDEO CODING

BACKGROUND OF THE INVENTION

This invention relates to video coding.

A video sequence consists of a series of still pictures or frames. Video compression methods are based on reducing the redundant and perceptually irrelevant parts of video sequences. The redundancy in video sequences can be categorised into spectral, spatial and temporal redundancy. Spectral redundancy refers to the similarity between the different colour components of the same picture. Spatial redundancy results from the similarity between neighbouring pixels in a picture. Temporal redundancy exists because objects appearing in a previous image are also likely to appear in the current image. Compression can be achieved by taking advantage of this temporal redundancy and predicting the current picture from another picture, termed an anchor or reference picture. Further compression is achieved by generating motion compensation data that describes the motion between the current picture and the previous picture.

However, sufficient compression cannot usually be achieved by only reducing the inherent redundancy of the sequence. Thus, video encoders also try to reduce the quality of those parts of the video sequence which are subjectively less important. In addition, the redundancy of the encoded bit-stream is reduced by means of efficient lossless coding of compression parameters and coefficients. The main technique is to use variable length codes.

Video compression methods typically differentiate between pictures that utilise temporal redundancy reduction and those that do not. Compressed pictures that do not utilise temporal redundancy reduction methods are usually called INTRA or I-frames or I-pictures. Temporally predicted images are usually forwardly predicted from a picture occurring before the current picture and are called INTER or P-frames. In the INTER frame case, the predicted motion-compensated picture is rarely precise enough and therefore a spatially compressed prediction error frame is associated with each INTER frame. INTER pictures may contain INTRA-coded areas.

Many video compression schemes also use temporally bi-directionally predicted frames, which are commonly referred to as B-pictures or B-frames. B-pictures are inserted between anchor picture pairs of I- and/or P-frames and are predicted from either one or both of these anchor pictures. B-pictures normally yield increased compression as compared with forward-predicted pictures. B-pictures are not used as anchor pictures, i.e., other pictures are not predicted from them. Therefore they can be discarded (intentionally or unintentionally) without impacting the picture quality of future pictures. Whilst B-pictures may improve compression performance as compared with P-pictures, their generation requires greater computational complexity and memory usage, and they introduce additional delays. This may not be a problem for non-real time applications such as video streaming but may cause problems in real-time applications such as video-conferencing.

A compressed video clip typically consists of a sequence of pictures, which can be roughly categorised into temporally independent INTRA pictures and temporally differentially coded INTER pictures. Since the compression efficiency in INTRA pictures is normally lower than in INTER pictures, INTRA pictures are used sparingly, especially in low bit-rate applications.

A video sequence may consist of a number of scenes or shots. The picture contents may be remarkably different

2

from one scene to another, and therefore the first picture of a scene is typically INTRA-coded. There are frequent scene changes in television and film material, whereas scene cuts are relatively rare in video conferencing. In addition, INTRA pictures are typically inserted to stop temporal propagation of transmission errors in a reconstructed video signal and to provide random access points to a video bit-stream.

Compressed video is easily corrupted by transmission errors, mainly for two reasons. Firstly, due to utilisation of temporal predictive differential coding (INTER frames), an error is propagated both spatially and temporally. In practice this means that, once an error occurs, it is easily visible to the human eye for a relatively long time. Especially susceptible are transmissions at low bit-rates where there are only a few INTRA-coded frames, so temporal error propagation is not stopped for some time. Secondly, the use of variable length codes increases the susceptibility to errors. When a bit error alters the codeword, the decoder will lose codeword synchronisation and also decode subsequent error-free codewords (comprising several bits) incorrectly until the next synchronisation (or start) code. A synchronisation code is a bit pattern which cannot be generated from any legal combination of other codewords and such codes are added to the bit stream at intervals to enable re-synchronisation. In addition, errors occur when data is lost during transmission. For example, in video applications using the unreliable UDP transport protocol in IP networks, network elements may discard parts of the encoded video bit-stream.

There are many ways for the receiver to address the corruption introduced in the transmission path. In general, on receipt of a signal, transmission errors are first detected and then corrected or concealed by the receiver. Error correction refers to the process of recovering the erroneous data perfectly as if no errors had been introduced in the first place. Error concealment refers to the process of concealing the effects of transmission errors so that they are hardly visible in the reconstructed video sequence. Typically some amount of redundancy is added by the source or transport coding in order to help error detection, correction and concealment. Error concealment techniques can be roughly classified into three categories: forward error concealment, error concealment by post-processing and interactive error concealment. The term "forward error concealment" refers to those techniques in which the transmitter side adds redundancy to the transmitted data to enhance the error resilience of the encoded data. Error concealment by post-processing refers to operations at the decoder in response to characteristics of the received signals. These methods estimate the correct representation of erroneously received data. In interactive error concealment, the transmitter and receiver co-operate in order to minimise the effect of transmission errors. These methods heavily utilise feedback information provided by the receiver. Error concealment by post-processing can also be referred to as passive error concealment whereas the other two categories represent forms of active error concealment.

There are numerous known concealment algorithms, a review of which is given by Y. Wang and Q.-F. Zhu in "Error Control and Concealment for Video Communication: A Review", Proceedings of the IEEE, Vol. 86, No. 5, May 1998, pp. 974-997 and an article by P. Salama, N. B. Shroff, and E. J. Delp, "Error Concealment in Encoded Video," submitted to IEEE Journal on Selected Areas in Communications.

Current video coding standards define a syntax for a self-sufficient video bit-stream. The most popular standards at the time of writing are ITU-T Recommendation H.263,

"Video coding for low bit rate communication", February 1998; ISO/IEC 14496-2, "Generic Coding of Audio-Visual Objects. Part 2: Visual", 1999 (known as MPEG-4); and ITU-T Recommendation H.262 (ISO/IEC 13818-2) (known as MPEG-2). These standards define a hierarchy for bit-streams and correspondingly for image sequences and images.

In H.263, the syntax has a hierarchical structure with four layers: picture, picture segment, macroblock, and block layer. The picture layer data contain parameters affecting the whole picture area and the decoding of the picture data. Most of this data is arranged in a so-called picture header.

The picture segment layer can either be a group of blocks layer or a slice layer. By default, each picture is divided into groups of blocks. A group of blocks (GOB) typically comprises 16 successive pixel lines. Data for each GOB consists of an optional GOB header followed by data for macroblocks. If the optional slice structured mode is used, each picture is divided into slices instead of GOBs. A slice contains a number of successive macroblocks in scan-order. Data for each slice consists of a slice header followed by data for the macroblocks.

Each GOB or slice is divided into macroblocks. A macroblock relates to 16x16 pixels (or 2x2 blocks) of luminance and the spatially corresponding 8x8 pixels (or block) of chrominance components. A block relates to 8x8 pixels of luminance or chrominance.

Block layer data consists of uniformly quantised discrete cosine transform coefficients, which are scanned in zigzag order, processed with a run-length encoder and coded with variable length codes. MPEG-2 and MPEG-4 layer hierarchies resemble the one in H.263.

By default these standards generally use the temporally previous reference picture (I or P) (also known as an anchor picture) as a reference for motion compensation. This piece of information is not transmitted, i.e., the bit-stream does not include information identifying the reference picture. Consequently, decoders have no means to detect if a reference picture is lost. Although many transport coders place video data into packets and associate a sequence number with the packets, these sequence numbers are not related to the video bit-stream. For example, a section of video bit-stream may contain P-picture P1, B-picture B2, P-picture P3, and P-picture P4, captured (and to be displayed) in this order. However, this section would be compressed, transmitted, and decoded in the following order: P1, P3, B2, P4 since B2 requires both P1 and P3 before it can be encoded or decoded.

Assuming that there is one picture per packet, that each packet contains a sequence number and that the packet carrying B2 is lost, the receiver can detect this packet loss from the packet sequence numbers. However, the receiver has no means to detect if it has lost a motion compensation reference picture for P4 or if it has lost a B-picture, in which case it could continue decoding normally.

The decoder therefore usually sends an INTRA request to the transmitter and freezes the picture on the display. However the transmitter may not be about to respond to this request. For instance in a non-real-time video streaming application, the transmitter cannot respond to an INTRA request from a decoder. Therefore the decoder freezes the picture until the next INTRA frame is received. In a real-time application such as video-conferencing, the transmitter may not be able to respond. For instance, in a multi-party conference, the encoder may not be able to respond to individual requests. Again the decoder freezes the picture until an INTRA frame is output by the transmitter.

SUMMARY OF THE INVENTION

According to a first aspect of the invention there is provided a method of encoding a video signal representing a sequence of pictures, the method employing both non-temporal prediction and temporal prediction, wherein the method comprises, for each picture that forms a reference picture for the temporal prediction of another picture, associating with each such picture an indicator indicating the temporal order of the reference picture in the encoded video signal relative to the other reference pictures in the encoded video signal.

Thus each reference picture (e.g. I-frames and P-frames) is associated with a sequence number. Preferably the indicator is incremented each time a reference picture is encoded. Most advantageously the indicator is incremented by one each time a reference picture is encoded. Thus the indicator is incremented by one from the previous reference picture.

If multi-layer coding is used, preferably this indicator is incremented by one from the previous reference picture in the same enhancement layer.

Including this indicator means that a decoder is capable of determining whether a reference picture has been lost and to take appropriate action, if available. This is the case even if the transport protocol does not include sequence information about the packets being transmitted or the transmitter uses a varying encapsulation strategy. For example, if the transmitter encapsulates a varying number of GOBs in one packet, receivers cannot reliably know how many GOBs or entire pictures were lost even if they could detect packet losses from packet sequence numbers.

The invention also enables a decoder to differentiate B picture losses from reference picture losses. Consequently, decoders can continue decoding after a B picture loss instead of waiting for the next INTRA picture.

In addition a decoder may continue decoding lower enhancement layers if a reference picture from a higher enhancement layer is lost.

The reference picture order number may be in respect of the whole picture or part of a picture. In the former case, typically the reference picture order number is included in a picture header. In a preferred implementation of the invention, the video signal is encoded according to the H.263 standard and the indicator is included in the Supplemental Enhancement Information. Where the RPON is in respect of part of a picture the reference picture order number may be included in the picture segment headers or macroblock headers of the encoded picture.

In accordance with a second aspect of the invention there is provided a method of decoding an encoded video signal representing a sequence of pictures, the method comprising receiving an encoded video signal, decoding each received picture, examining for each picture to be decoded that forms a reference picture for another picture an indicator representing the temporal order of a reference frame and, when the indicator does not follow consecutively from an immediately preceding decoded reference frame, detecting a lost reference frame.

Preferably the decoder sends a request to a transmitter to encode a picture in a non-temporal manner when the indicator does not follow consecutively from an immediately preceding decoded reference frame.

In accordance with a third aspect of the invention there is provided an encoded video signal wherein for each picture that forms a reference picture for the temporal prediction of

5

another picture, an indicator is associated with each such reference picture indicating the temporal order of the reference pictures in the encoded video stream.

In accordance with a fourth aspect of the invention there is provided a video encoder comprising an input for receiving a video signal representing a sequence of pictures and for generating encoded pictures, said encoder being arranged to employ both non-temporal prediction and temporal prediction, wherein the encoder is arranged, for each picture that forms a reference picture for the temporal prediction of another picture, to associate with each reference picture an indicator indicating the temporal order of the reference picture in the encoded video signal relative to other reference pictures in the encoded video signal.

Preferably the indicator is incremented each time a reference picture is encoded.

In accordance with a fifth aspect of the invention there is provided a video decoder comprising an input for receiving an encoded video signal representing a sequence of pictures, a decoder for decoding each received picture, the decoder being arranged to examine for each picture to be decoded that forms a reference picture for another picture an indicator representing the temporal order of a reference frame and, when the indicator does not follow consecutively from an immediately preceding decoded reference frame, to detect a lost reference frame.

The invention also relates to a portable radio communications device including an encoder and/or a decoder as described.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will now be described, by way of example only, with reference to the accompanying drawings, in which:

FIG. 1 shows a multimedia mobile communications system;

FIG. 2 shows an example of the multimedia components of a multimedia terminal;

FIG. 3 shows an example of a video codec;

FIG. 4 illustrates the operation of a first implementation of a video encoder according to a first embodiment of the invention;

FIG. 5 illustrates the operation of a second implementation of a video encoder according to a first embodiment of the invention;

FIG. 6 shows the syntax of a bit stream as known according to H.263;

FIG. 7 shows a first example of a bit stream output by an encoder according to the invention;

FIG. 8 shows a second example of a bit stream output by an encoder according to the invention;

FIG. 9 shows a third example of a bit stream output by an encoder according to the invention;

FIG. 10 illustrates enhancement layers used in video coding for SNR scalability; and

FIG. 11 illustrates enhancement layers used in video coding for spatial scalability.

DETAILED DESCRIPTION OF THE INVENTION

FIG. 1 shows a typical multimedia mobile communications system. A first multimedia mobile terminal 1 communicates with a second multimedia mobile terminal 2 via a

6

radio link 3 to a mobile communications network 4. Control data is sent between the two terminals 1,2 as well as the multimedia data.

FIG. 2 shows the typical multimedia components of a terminal 1. The terminal comprises a video codec 10, an audio codec 20, a data protocol manager 30, a control manager 40, a multiplexer/demultiplexer 50 and a modem 60 (if the required). The video codec 10 receives signals for coding from a video capture device of the terminal (not shown) (e.g. a camera) and receives signals for decoding from a remote terminal 2 for display by the terminal 1 on a display 70. The audio codec 20 receives signals for coding from the microphone (not shown) of the terminal 1 and receive signals for decoding from a remote terminal 2 for reproduction by a speaker (not shown) of the terminal 1.

The control manager 40 controls the operation of the video codec 10, the audio codec 20 and the data protocol manager 30. However, since the invention is concerned with the operation of the video codec 10, no further discussion of the audio codec 20 and protocol manager 30 will be provided.

FIG. 3 shows an example of a video codec 10 according to the invention. The video codec comprises an encoder part 100 and a decoder part 200. The encoder part 100 comprises an input 101 for receiving a video signal from a camera or video source (not shown) of the terminal 1. A switch 102 switches the encoder between an INTRA-mode of coding and an INTER-mode.

In INTRA-mode, the video signal from the input 101 is transformed into DCT coefficients by a DCT transformer 103. The DCT coefficients are then passed to a quantiser 104 that quantises the coefficients. Both the switch 102 and the quantiser 104 are controlled by an encoding control manager 105 of the video codec, which also receives feedback control from the receiving terminal 2 by means of the control manager 40.

In INTER mode, the switch 102 is operated to accept from a subtractor 106 the difference between the signal from the input 101 and a previous picture which is stored in a picture store 107. The difference data output from the subtractor 106 represents the prediction error between the current picture and the previous picture stored in the picture store 107. The data in the picture store 107 is generated by passing the data output by the quantiser through an inverse quantiser 108 and applying an inverse DCT transform 109 to the inverse-quantised data. The resulting data is added to the contents of the picture store 107 by adder 110. A motion estimator 111 may generate motion compensation data from the data in the picture store 107 in a conventional manner.

The encoding control manager 105 decides whether to apply INTRA or INTER coding or whether to code the frame at all on the basis of either the output of the subtractor 106 or in response to feedback control data received a receiving decoder. When not responding to feedback control data, the encoder typically encodes a frame as an INTRA-frame either only at the start of coding (all other frames being P-frames), or at regular periods e.g. every 5s, or when the output of the subtractor exceeds a threshold i.e. when the current picture and that stored in the picture store 107 are too dissimilar. The encoder may also be programmed to encode frames in a particular regular sequence e.g. I B B P B B P B B P B B P B B I B B P etc. In addition the encoding control manager may decide not to code a received frame at all. This happens when the similarity between the current frame and the reference frame is so high that the encoder decides not to encode the current frame. The encoding control manager operates the switch accordingly.

7

The video codec outputs the quantised DCT coefficients **112a**, the quantising index **112b** (i.e. the details of the quantiser used), an INTRA/INTER flag **112c** to indicate the mode of coding performed (I or P/B), a transmit flag **112d** to indicate the number of the frame being coded and the motion vectors **112e** for the picture being coded. These are multiplexed together by the multiplexer **50** together with other multimedia signals.

The decoder part **200** of the video codec **10** comprises an inverse quantiser **120**, an inverse DCT transformer **121**, a motion compensator **122**, a picture store **123** and a controller **124**. The controller **124** receives video codec control signals demultiplexed from the encoded multimedia stream by the demultiplexer **50**. In practice the controller **105** of the encoder and the controller **124** of the decoder may be the same processor.

The operation of an encoder according to the invention will now be described. The video codec **10** receives a video signal to be encoded. The encoder **100** of the video codec encodes the video signal by performing DCT transformation, quantisation and motion compensation. The decoded video data is then output to the multiplexer **50**. The multiplexer **50** multiplexes the video data from the video codec **10** and control data from the control manager **40** (as well as other signals as appropriate) into a multimedia signal. The terminal **1** outputs this multimedia signal to the receiving terminal **2** via the modem **60** (if required).

In a first embodiment of the invention, each time the encoder encodes a frame which may form the reference frame for a subsequent frame, the encoding control manager **105** associates with the frame a so-called Reference Picture Order Number (RPON). For example, a RPON is associated with every I or P frame of a video signal but not with a B-frame. The RPON value is incremented each time a successive reference picture is encoded, preferably by 1.

The encoding control manager **105** outputs the RPON codeword on output **112f** which indicates the Reference Picture Order Number associated with the encoded frame. This is multiplexed into the video bitstream by a multiplexer.

FIG. **4** illustrates the operation of the encoder. In this embodiment, the encoder is arranged to output an I-frame when the similarity between the frame being coded and the reference frame is less than a first threshold i.e. when the output from the subtractor **106** is greater than a first threshold. Otherwise the encoder outputs P-frames. The first line of FIG. **4** represents the frames of data received from a capture input device and input to the video encoder on input **101**. The second line of FIG. **4** represents those frames of the input signal that the encoder decides to encode and the coding mode used to encode each frame. As mentioned above some, the encoding control manager may decide that a frame is not to be coded: this is exemplified in FIG. **4** by the fact that frames **2**, **3** and **11** are not coded.

Frame **1** is coded in INTRA-mode; frame **4** is encoded as a P-frame with reference to frame **1**; frame **5** is encoded as a P-frame with reference to frame **4**; frame **6** is encoded as a P-frame with reference to frame **5**; frame **7** is encoded as a P-frame with reference to frame **6**; frame **8** is encoded as an I-frame; frame **9** is encoded as a P-frame with reference to frame **8**; frame **10** is encoded as a P-frame with reference to frame **9**; frame **12** is encoded as a P-frame with reference to frame **10**.

In this embodiment all (but the last) of the encoded frames act as the reference frame for a later frame. Thus a RPON is associated with all of the frames to be coded, as shown in the third line of FIG. **4**. As can be seen, the RPON is incremented by 1 each time.

8

The fourth line of FIG. **4** shows the Temporal Reference (TR) of the encoded frame. This is a field included in H.263 and the value of TR is formed by incrementing its value in the temporally previous reference picture header by one plus the number of skipped or non-reference pictures since the previously transmitted one. Thus in the example shown in FIG. **4** the TR shown for each frame is the same as the original number in the original signal input to **102**.

FIG. **5** shows a second embodiment of an encoder according to the invention. In this embodiment, the encoder is arranged to code the frames according to the regular sequence I B B P B B P B B P B B P B I B B P. The first line of FIG. **5** shows the input frames and the second line shows the coded frames and their coding mode, I, P or B.

The frames are received from a video capture device in the order **1,2,3,4,5,6** etc. and are displayed in this order i.e. the decoded frames are displayed in the order **I1, B2, B3, P4, B5, B6, P7** etc. However the video bit stream is compressed, transmitted and decoded in the following order **I1, P4, B2, B3, P7, B5, B6** etc. This is because each B-frame requires preceding and succeeding reference frames before they can be encoded/decoded i.e. frame **B2** requires frame **I1** and **P4** to be encoded/decoded before frame **B2** can be encoded/decoded.

As explained previously, B-frames are inserted between anchor picture pairs of I- and/or P-frames and are predicted from either one or both of these anchor pictures. Thus in the illustration given in FIG. **5**, Frame **1** is coded in INTRA-mode; frame **4** is encoded as a B-frame with reference to frame **1** and/or **6**; frame **5** is encoded as a B-frame with reference to frame **1** and/or **6**; frame **6** is encoded as a P-frame with reference to frame **1**; frame **7** is encoded as a B-frame with reference to frame **6** and/or **9**; frame **8** is encoded as an B-frame with reference to frame **6** and/or **9**; frame **9** is encoded as a P-frame with reference to frame **6**; frame **10** is encoded as a B-frame with reference to frame **9** and/or **13** (not shown); frame **12** is encoded as a B-frame with reference to frame **9** and/or **13** and so on.

In this embodiment each I-frame and P-frame of the encoded sequence acts as a reference frame for another frame. However a B-frame does not act as a reference picture for any other frame. Thus a RPON is associated with all of the I-frames and P-frames, as shown in the third line of FIG. **5**. As can be seen, the RPON is incremented by 1 each time. Thus frame **1** (an I-frame) has a RPON of 1, frame **4** (a P-frame) has a RPON of 2 and frame **9** (a P-frame) has a RPON of 3.

The fourth line of FIG. **5** shows the Temporal Reference (TR) of the encoded frame. As in the example shown in FIG. **4**, the TR shown for each frame is the same as the order of occurrence in the original signal input to **10**.

Considering the terminal **1** as receiving coded video data from terminal **2**, the operation of the video codec **10** will now be described with reference to its decoding role. The terminal **1** receives a multimedia signal from the transmitting terminal **2**. The demultiplexer **50** demultiplexes the multimedia signal and passes the video data to the video codec **10** and the control data to the control manager **40**. The decoder **200** of the video codec decodes the encoded video data by inverse quantising, inverse DCT transforming and motion compensating the data. The controller **124** of the decoder checks the integrity of the received data and, if an error is detected, attempts to conceal the error in a manner to be described below. The decoded, corrected and concealed video data is then output for reproduction on a display **70** of the receiving terminal **1**.

Errors in video data may occur at the picture level, the GOB level or the macroblock level. Error checking may be carried out at any or all of these levels.

Considering first the signal as shown in FIG. 4, when a decoder according to the invention receives this signal each frame of the signal is decoded in a conventional manner and then displayed on a display means. The decoded frame may be error corrected and error concealed in a conventional manner. Each time a frame is decoded, the decoder examines the TR field to determine when the frame is to be displayed. If the TRs are not consecutive (e.g. the decoder receives a frame with TR=1 and then a frame with TR=4) the decoder holds the frame 1 on the display for 3 times the usual frame period, as is conventional. The decoder also examines the RPON of the received frames. In the case shown in FIG. 4 the decoder receives frame 1 and sees that this frame has a RPON=1; the decoder then receives a frame with TR=4 and RPON=2. The decoder compares the RPON of the currently received frame with the RPON of the previously received frame and calculates the difference between the RPON values. In this case the difference is 1 and the decoder therefore knows that no reference pictures have been lost between the current frame and the previous decoded reference frame. The decoder therefore continues to decode the signal in a conventional manner.

Let us now assume that the decoder is unable to reconstruct frame 5 (this could be due to the data being greatly corrupted or being lost altogether) and the next frame received and decoded by the decoder is frame 6. The decoder compares the RPON of the currently received frame (frame 6) with the RPON of the previously received and decoded reference frame (frame 4) and calculates the difference between the RPON values. In this case the difference is 2 and the decoder therefore knows that a reference picture has been lost between transmission of the current frame and that of the previous frame. If the decoder has the facility to send control feedback data to the transmitting video encoder the decoder can send a request to the transmitting video encoder to encode a frame as an INTRA-frame and so stop the temporal error propagation that would result from frame 6 being decoded with reference to frame 4.

Considering now the signal as shown in FIG. 5, when a decoder according to the invention receives this signal each frame of the signal is decoded in a conventional manner and then displayed on a display means. The decoded frame may be error corrected and error concealed in a conventional manner. Each time a frame is decoded, the decoder examines the TR field to determine when the frame is to be displayed. The decoder also examines the RPON of the received frames.

In the case shown in FIG. 5 the decoder receives frame 1 and sees that this frame has a RPON=1. The decoder decodes this frame in a conventional INTRA-mode manner. The next frame received by the decoder is then frame 6, with TR=6 and RPON=2. The decoder compares the RPON of the currently received frame (frame 6) with the RPON of the previously received and decoded reference frame (frame 1) and calculates the difference between the RPON. In this case the difference is 1 and the decoder therefore knows that no reference pictures have been lost between transmission of the current frame and that of the previous decoded reference frame. The decoder then decodes frame 6 with reference to frame 1.

The decoder then receives a frame with TR=4 and no RPON. In this case the decoder makes no further use of the RPON and decodes frame 4 with reference to decoded frames 1 and 6.

Let us now assume that the decoder is unable to reconstruct frame 5 (this could be due to the data being greatly corrupted or being lost altogether). The fact that B-frame 5 has been lost is of no consequence to the decoder as the B-frame does not form a reference picture for any other frame and thus its loss will not introduce any temporal error propagation.

The next frame to be received is frame 9. However, let us now assume that the decoder is unable to reconstruct frame 9, which is a P-frame (this could be due to the data being greatly corrupted or being lost altogether). The decoder may therefore be unable to decode successfully any of frames 7, 8, 10 or 12 since these may all be predicted, in part at least, with reference to frame 9. Typically, in this situation, the decoder will freeze the displayed picture.

The next frame received and decoded by the decoder is frame 13. The decoder compares the RPON of the currently received reference frame (frame 13) with the RPON of the previously received and decoded reference frame (frame 6) and calculates the difference between the RPON. In this case the difference is 2 and the decoder therefore knows that a reference picture has been lost between the current frame and the previous decoded reference frame. If the decoder has the facility to send control feedback data to the transmitting video encoder the decoder can send a request to the transmitting video encoder to encode a frame as an INTRA-frame and so stop the temporal error propagation that would result from frame 13 being decoded with reference to frame 6.

How the reference picture order number may be included in the encoded signal will now be addressed with reference to the H.263 video coding standard.

FIG. 6 shows the syntax of a bit stream as known according to H.263. The following implementations describe the GOB format but it will be clear to a skilled person that the invention may also be implemented in the slice format.

As mentioned already, the bit stream has four layers: the picture layer, picture segment layer, macroblock layer and block layer. The picture layer comprises a picture header followed by data for the Group of Blocks, eventually followed by any optional end-of-sequence code and stuffing bits.

The prior art H.263 bit stream is formatted as shown in FIG. 6. A descriptor for each part is given below:

PSC	The picture start code (PSC) indicates the start of the picture
TR	The Temporal Reference (TR) is formed by incrementing its value in the temporally previous reference picture header by one plus the number of skipped or non-referenced pictures since the previously transmitted one
PTYPE	Amongst other things, PTYPE includes details of the picture coding type i.e. INTRA or INTER
PQUANT	A codeword that indicates the quantiser to be used for the picture until updated by any subsequent quantiser information
CPM	A codeword that signals the use of optional continuous presence multipoint and video multiplex (CPM) mode
PSBI	Picture Sub-Bit stream Indicator-only present if CPM is set
TR _B	Present if the frame is a bi-directionally predicted frame (known as a PB-frame)
DBQUANT	Present if a bi-directional frame
PEI	This relates to extra insertion information and is set to "1" to indicate the presence of the following optional data fields PSUPP and PEI. PSUPP and PEI are together known as Supplemental Enhancement Information, which is further defined in Annex L of H263.

-continued

GOBS	Is the data for the group of blocks for the current picture
ESTF	A stuffing codeword provided to attain byte alignment before EOS
EOS	A codeword indicating the end of the data sequence for the picture
PSTUF	A stuffing codeword to allow for byte alignment of the next picture start code PSC

The structure as shown in FIG. 4 does not include the optional PLUSTYPE data field. PSBI is only present if indicated by CPM. TRB and DBQUANT are only present if PTYPE indicates use of a so-called PB frame mode (unless the PLUSTYPE field is present and the use of DBQUANT is indicated therein). These issues are addressed in more detail in the H.263 specification.

The following paragraphs outline possible implementations of the bit-stream output by an encoder according to the first aspect of the invention.

The reference picture order number may be incorporated into a H.263 bit stream as follows. FIG. 7 shows an example of a bit stream output by an encoder according to the first implementation of the invention. As shown in FIG. 7, the bit stream includes a further codeword RPON which is a codeword indicating the reference picture order number. This is inserted by an encoder according to the invention, as described above.

Alternatively, the reference picture order number may be included in the Supplemental Enhancement Information PSUPP (see Annex L of H.263 and FIG. 4). The supplemental information may be present in the bit stream even though the decoder may not be capable of providing the enhanced capability to use it, or even to properly interpret it. Simply discarding the supplemental information is allowable by decoders unless a requirement to provide the requested capability has been negotiated by the transmitter and receiver.

If PEI is set to "1", then 9 bits follow consisting of 8 bits of data (PSUPP) and then another PEI bit to indicate if a further 9 bits follow and so on.

The PSUPP data consists of a 4-bit function type indication FTYPE, followed by a 4-bit parameter data size specification DSIZE followed by DSIZE octets of function parameter data, optionally followed by another FTYPE and so on. It is known to use this PSUPP codeword to signal various situations such as: to indicate a full-picture or partial-picture freeze or freeze-release request with or without resizing; to tag particular pictures or sequences of pictures within the video stream for external use; or to convey chroma key information for video compositing.

To implement the invention using the Supplemental Enhancement Information, a further FTYPE is defined as Reference Picture Order Number.

FIG. 8 illustrates the example where a parameter RPON is included in the SEI of the picture header. The FTYPE is defined as RPON. The DSIZE specifies the size of the parameter and the following octet is the parameter data i.e. the value of RPON. From this value a receiving decoder can determine whether a reference picture has been lost.

Alternatively, the information may be contained in the additional Supplemental Enhancement Information as specified in a "Draft of new Annex W: Additional Supplementary Enhancement Information Specification" P. Ning and S. Wenger, ITU-T Study Group 16 Question 15 Document Q15-I58, November 1999.

In this draft proposal, FTYPE 14 is defined as "Picture Message". When this FTYPE is set, the picture message function indicates the presence of one or more octets representing message data. The first octet of the message data is a message header with the structure shown in FIG. 9 i.e. CONT, EBIT and MTYPE. DSIZE is equal to the number of octets in the message data corresponding to a picture message function, including the first octet message header.

The continuation field CONT, if equal to 1, indicates that the message data associated with the picture message is part of the same logical message as the message data associated with the next picture message function. The End Bit Position field EBIT specifies the number of least significant bits that shall be ignored in the last message octet. Further details of these fields can be found in the draft of Annex W referred to above.

The field MTYPE indicates the type of message. Various types of message are suggested in the draft of Annex W. According to the invention one type e.g. MTYPE 12 is defined as RPON or Picture Number. The message contains two data bytes that carry a 10-bit Picture Number. Consequently, DSIZE shall be 3, CONT shall be 0, and EBIT shall be 6. Picture Number shall be incremented by 1 for each coded and transmitted I or P picture or PB or Improved PB frame, in a 10-bit modulo operation. For EI and EP pictures, Picture Number shall be incremented for each EI or EP picture within the same scalability enhancement layer. For B pictures, Picture Number shall be incremented relative to the value in the most recent non-B picture in the reference layer of the B picture which precedes the B picture in bitstream order (a picture which is temporally subsequent to the B picture). If adjacent pictures in the same enhancement layer have the same temporal reference, and if the reference picture selection mode (see Annex N) is in use, the decoder shall regard this occurrence as an indication that redundant copies have been sent of approximately the same pictured scene content, and all of these pictures shall share the same Picture Number. If the difference (modulo 1024) of the Picture Numbers of two consecutively received non-B pictures in the same enhancement layer is not 1, and if the pictures do not represent approximately the same pictured scene content as described above, a loss of pictures or corruption of data may be inferred by the decoder. The value of RPON is defined in the octet following the message header.

In a specific example, this message contains one data byte, i.e., DSIZE is 2, CONT is 0, and EBIT is 0.

The Reference Picture Order Number is incremented by one from the corresponding number of the previous coded reference picture. The least significant 8-bits of the result of the incrementation is placed in the data byte associated with this message.

The invention may also be implemented in accordance with Annex U of H.263.

The above description has made reference to encoded video streams in which bi-directionally predicted pictures (B-pictures) are encoded. As mentioned earlier, B-pictures are never used as reference pictures. Since they can be discarded without impacting the picture quality of future pictures, they provide temporal scalability. Scalability allows for the decoding of a compressed video sequence at more than one quality level. In other words, a scalable multimedia clip can be compressed so that it can be streamed over channels with different data rates and still be decoded and played back in real-time.

Thus the video stream may be decoded in different ways by differing decoders. For instance, a decoder can decide

only to decode the I- and P-pictures of a signal, if this is the maximum rate of decoding that the decoder can attain. However if a decoder has the capacity, it can also decode the B-pictures and hence increase the picture display rate. Thus the perceived picture quality of the displayed picture will be enhanced over a decoder that only decodes the I- and P-pictures.

Scalable multimedia is typically ordered so that there are hierarchical layers of data. A base layer contains a basic representation of the multimedia clip whereas enhancement layers contain refinement data on top of underlying layers. Consequently, the enhancement layers improve the quality of the clip.

Scalability is a desirable property for heterogeneous and error prone environments. This property is desirable in order to counter limitations such as constraints on bit rate, display resolution, network throughput, and decoder complexity.

Scalability can be used to improve error resilience in a transport system where layered coding is combined with transport prioritisation. The term transport prioritisation here refers to various mechanisms to provide different qualities of service in transport, including unequal error protection, to provide different channels having different error/loss rates. Depending on their nature, data are assigned differently. For example, the base layer may be delivered through a channel with a high degree of error protection, and the enhancement layers may be transmitted through more error-prone channels.

Generally, scalable multimedia coding suffers from a worse compression efficiency than non-scalable coding. In other words, a multimedia clip encoded as a scalable multimedia clip with enhancement layers requires greater bandwidth than if it had been coded as a non-scalable single-layer clip with equal quality. However, exceptions to this general rule exist, for example the temporally scalable B-frames in video compression.

The invention may be applied to other scalable video compression systems. For instance, in H.263 Annex O, two other forms of scalability are defined: signal-to-noise (SNR) scalability and spatial scalability. Spatial scalability and SNR scalability are closely related, the only difference being the increased spatial resolution provided by spatial scalability. An example of SNR scalable pictures is shown in FIG. 10. SNR scalability implies the creation of multi-rate bit streams. It allows for the recovery of coding errors, or differences between an original picture and its reconstruction. This is achieved by using a finer quantiser to encode the difference picture in an enhancement layer. This additional information increases the SNR of the overall reproduced picture.

Spatial scalability allows for the creation of multi-resolution bit streams to meet varying display requirements and/or constraints. A spatially scalable structure is illustrated in FIG. 11. It is essentially the same as in SNR scalability except that a spatial enhancement layer attempts to recover the coding loss between an up-sampled version of the reconstructed reference layer picture and a higher resolution version of the original picture. For example, if the reference layer has a quarter common intermediate format (OCIF) resolution, and the enhancement layer has a common intermediate format (CIF) resolution, the reference layer picture must be scaled accordingly such that the enhancement layer picture can be predicted from it. The OCIF standard allows the resolution to be increased by a factor of two in the vertical direction only, horizontal direction only, or both the vertical and horizontal directions for a single enhancement

layer. There can be multiple enhancement layers, each increasing the picture resolution over that of the previous layer. The interpolation filters used to up-sample the reference layer picture are explicitly defined in the H.263 standard. Aside from the up-sampling process from the reference to the enhancement layer, the processing and syntax of a spatially scaled picture are identical to those of an SNR scaled picture.

In either SNR or spatial scalability, the enhancement layer pictures are referred to as EI- or EP-pictures. If the enhancement layer picture is upwardly predicted from a picture in the reference layer, then the enhancement layer picture is referred to as an Enhancement-I (EI) picture. In this type of scalability, the reference layer means the layer "below" the current enhancement layer. In some cases, when reference layer pictures are poorly predicted, over-coding of static parts of the picture can occur in the enhancement layer, causing an unnecessarily excessive bit rate. To avoid this problem, forward prediction is permitted in the enhancement layer. A picture that can be predicted in the forward direction from a previous enhancement layer picture or, alternatively, upwardly predicted from the reference layer picture is referred to as an Enhancement-P (EP) picture. Note that computing the average of the upwardly and forwardly predicted pictures can provide bi-directional prediction for EP-pictures. For both EI- and EP-pictures, upward prediction from the reference layer picture implies that no motion vectors are required. In the case of forward prediction for EP-pictures, motion vectors are required.

According to the invention, if the encoder is capable of multi-layer coding (for example as discussed in Annex O of H.263) the reference pictures of each layer are given consecutive Reference Picture Order Numbers. These may be associated with the enhancement layer number (ELNUM) of the current picture. The Reference Picture Order Number is incremented by one from the corresponding number of the previous coded reference picture in the same enhancement layer.

As shown in FIGS. 10 and 11, the pictures of the enhancement layer may be predicted from a preceding picture of the enhancement layer and/or from the equivalent I- or P-picture of the base layer. The enhancement layer may not be predicted from a B-picture in the reference layer.

If adjacent pictures in the same enhancement layer have the same temporal reference, and if Annex N or Annex U of H.263 is in use, the decoder preferably regards this occurrence as an indication that redundant copies have been sent of approximately the same pictured scene content, and all of these pictures then share the same RPON.

A decoder according to the invention, on receipt of a multi-layer signal as described above, attempts to decode the signal in a conventional manner. In each layer, each time a reference picture is decoded, the decoder examines the RPON of the decoded picture. If the decoder determines that a reference picture has been lost from an enhancement layer, the decoder ceases to display pictures from the enhancement layer until an EI-picture is received. The decoder continues to decode the base layer as described earlier.

The invention is not intended to be limited to the video coding protocols discussed above: these are intended to be merely exemplary. The invention is applicable to any video coding protocol in which temporal prediction may be used. The addition of the information as discussed above allows a receiving decoder to determine that a reference picture has been lost and to take appropriate action.

15

What is claimed is:

1. A method of encoding a video signal representing a sequence of pictures to form an encoded video signal comprising temporally independent INTRA pictures and temporally predicted pictures, wherein the INTRA pictures and at least some of the temporally predicted pictures are used to form reference pictures for the temporal prediction of other pictures in the video sequence, comprising indicating an encoding order of those pictures used to form reference pictures in the encoded video signal with a sequence indicator having an independent numbering scheme, such that consecutive pictures used to form reference pictures in encoding order are assigned sequence indicator values that differ with respect to each other by a predetermined amount independent of the number of non-reference pictures encoded between successive reference pictures.

2. A method according to claim 1, wherein said predetermined amount is one.

3. A method according to claim 1, wherein the sequence indicator is included in a picture header.

4. A method according to claim 3, wherein the video signal is encoded according to the H.263 video coding standard and the sequence indicator is included in the Supplemental Enhancement Information of a bit stream of the H.263 video coding standard.

5. A method of decoding an encoded video signal representing a sequence of pictures to form a decoded video signal, the method comprising receiving an encoded video signal comprising temporally independent INTRA pictures and temporally predicted pictures, wherein the INTRA pictures and at least some of the temporally predicted pictures are used to form reference pictures for the temporal prediction of other pictures, the encoded video signal further comprising a sequence indicator having an independent numbering scheme such that consecutive reference pictures in encoding order are assigned sequence indicator values that differ with respect to each other by a predetermined amount independent of the number of non-reference pictures encoded between successive reference pictures, decoding received encoded pictures, examining each decoded picture that forms a reference picture to identify the sequence indicator value assigned to the reference picture and comparing the sequence indicator values assigned to consecutively decoded reference pictures to detect loss of a reference picture.

6. A method according to claim 5, further comprising sending a request to a transmitter to encode a picture in a non-temporally predicted manner when the sequence indicator value assigned to a particular reference picture does not follow consecutively from that associated with an immediately preceding decoded reference picture.

7. A video encoder comprising an input for receiving a video signal representing a sequence of pictures, the video encoder for generating an encoded video signal comprising temporally independent INTRA pictures and temporally predicted pictures, wherein the INTRA pictures and at least some of the temporally predicted pictures form reference pictures for temporal prediction of other pictures, and the encoder is arranged to indicate an encoding order of the reference pictures in the encoded video signal with a sequence indicator having an independent numbering scheme, such that consecutive reference pictures in encoding order are assigned sequence indicator values that differ with respect to each other by a predetermined amount independent of the number of non-reference pictures encoded between successive reference pictures.

16

8. A video encoder according to claim 7, wherein said predetermined amount is one.

9. A video decoder for decoding an encoded video signal representing a sequence of pictures to form a decoded video signal, the encoded video signal comprising temporally independent INTRA pictures and temporally predicted pictures, wherein the INTRA pictures and at least some of the temporally predicted pictures form reference pictures for the temporal prediction of other pictures, the encoded video signal further comprising a sequence indicator having an independent numbering scheme such that consecutive reference pictures in encoding order are assigned sequence indicator values that differ with respect to each other by a predetermined amount independent of the number of non-reference pictures encoded between successive reference pictures, comprising an input for receiving the encoded video signal and being arranged to decode received encoded pictures, to examine each decoded picture that forms a reference picture to identify the sequence indicator value assigned to the reference picture and to compare the sequence indicator values assigned to consecutively decoded reference pictures to detect loss of a reference picture.

10. A portable radio communications device including an encoder according to claim 7.

11. A method according to claim 1, wherein the sequence indicator is associated with a whole picture.

12. A method according to claim 1, wherein the sequence indicator is associated with part of a picture.

13. A method according to claim 12, wherein the sequence indicator is included in a picture segment header or a macroblock header of an encoded picture.

14. A method according to claim 1, wherein the video signal is scalably encoded and sequence indicators are associated with layers of the scalably encoded video signal.

15. A method according to claim 5, wherein the predetermined amount is one.

16. A method according to claim 5, wherein the sequence indicator is included in a picture header.

17. A method according to claim 5, wherein the video signal is encoded according to the H.263 video coding standard and the sequence indicator is included in the Supplemental Enhancement Information of a bit-stream of the H.263 video coding standard.

18. A method according to claim 5, wherein the sequence indicator is associated with a whole picture.

19. A method according to claim 5, wherein the sequence indicator is associated with part of a picture.

20. A method according to claim 19, wherein the sequence indicator is included in a picture segment header or a macroblock header of an encoded picture.

21. A method according to claim 5, wherein the video signal is scalably encoded and sequence indicators are associated with layers of the scalably encoded video signal.

22. A video encoder according to claim 7, arranged to include the sequence indicator in a picture header.

23. A video encoder according to claim 7, arranged to encode the video signal according to the H.263 video coding standard and to include the sequence indicator in the Supplemental Enhancement Information of a bit stream of the H.263 video coding standard.

24. A video encoder according to claim 7, arranged to associate the sequence indicator with a whole picture.

25. A video encoder according to claim 7, arranged to associate the sequence indicator with part of a picture.

26. A video encoder according to claim 7, arranged to include the sequence indicator in a picture segment header or a macroblock header of an encoded picture.

17

27. A video encoder according to claim 7, arranged to encode the video signal scalably and to associate sequence indicators with layers of the scalably encoded video signal.

28. A video decoder according to claim 9, further arranged to send a request to a transmitter to encode a picture in a non-temporally predicted manner when the sequence indicator value assigned to a particular reference picture does not follow consecutively from that associated with an immediately preceding decoded reference picture.

29. A video decoder according to claim 9, wherein the predetermined amount is one.

30. A video decoder according to claim 9, wherein the sequence indicator is included in a picture header.

31. A video decoder according to claim 9, wherein the video signal is encoded according to the H.263 video coding standard and the sequence indicator is included in the Supplemental Enhancement Information of a bit stream of the H.263 video coding standard.

32. A video decoder according to claim 9, wherein the sequence indicator is associated with a whole picture.

33. A video decoder according to claim 9, wherein the sequence indicator is associated with part of a picture.

34. A video decoder according to claim 33, wherein the sequence indicator is included in a picture segment header or a macroblock header of an encoded picture.

35. A video decoder according to claim 9, wherein the video signal is scalably encoded and sequence indicators are associated with layers of the scalably encoded video signal.

36. A multimedia terminal device including an encoder according to claim 7.

37. An encoded video signal representing a sequence of pictures, comprising temporally independent INTRA pictures and temporally predicted pictures, wherein the INTRA pictures and at least some of the temporally predicted pictures form reference pictures for the temporal prediction of other pictures, the encoded video signal further comprising a sequence indicator having an independent numbering

18

scheme for indicating an encoding order of the reference pictures, such that consecutive reference pictures in encoding order are assigned sequence indicator values that differ with respect to each other by a predetermined amount independent of the number of non-reference pictures encoded between successive reference pictures.

38. An encoded video signal according to claim 37, wherein said predetermined amount is one.

39. An encoded video signal according to claim 37, wherein the sequence indicator is included in a picture header.

40. An encoded video signal according to claim 37, encoded according to the H.263 video coding standard and the sequence indicator is included in the Supplemental Enhancement Information of a bit stream of the H.263 video coding standard.

41. An encoded video signal according to claim 37, wherein the sequence indicator is associated with the whole of a picture.

42. An encoded video signal according to claim 37, wherein the sequence indicator is associated with part of a picture.

43. An encoded video signal according to claim 42, wherein the sequence indicator is included in a picture segment header or a macroblock header of an encoded picture.

44. An encoded video signal according to claim 37, wherein the encoded video signal is scalably encoded and sequence indicators are associated with layers of the scalably encoded video signal.

45. A portable radio communications device including a decoder according to claim 9.

46. A multimedia terminal device including a decoder according to claim 9.

* * * * *

EXHIBIT 13



US008144764B2

(12) **United States Patent**
Hannuksela

(10) **Patent No.:** **US 8,144,764 B2**
(45) **Date of Patent:** ***Mar. 27, 2012**

(54) **VIDEO CODING**

(56) **References Cited**

(75) Inventor: **Miska Hannuksela**, Tampere (FI)

U.S. PATENT DOCUMENTS

(73) Assignee: **Nokia Oy**, Espoo (FI)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1249 days.

This patent is subject to a terminal disclaimer.

4,972,261	A	11/1990	Whalley
5,768,527	A	6/1998	Zhu et al.
6,111,915	A	8/2000	Fukunaga et al.
6,169,821	B1	1/2001	Fukunaga et al.
6,357,028	B1	3/2002	Zhu
6,393,152	B2	5/2002	Takahashi et al.
6,629,261	B1	9/2003	Chintada et al.
6,711,140	B1	3/2004	Agarwal et al.

FOREIGN PATENT DOCUMENTS

(21) Appl. No.: **11/242,888**

CN	1199531	A	11/1998
EP	0637027	A2	2/1995
EP	0702492	A1	3/1996
EP	0713340	A2	5/1996
EP	0 849 952		6/1998
EP	0849952		6/1998
EP	0 851 685		7/1998
EP	0851685		7/1998
GB	2268661		1/1994

(22) Filed: **Oct. 5, 2005**

(65) **Prior Publication Data**

US 2006/0029129 A1 Feb. 9, 2006

Related U.S. Application Data

(63) Continuation of application No. 09/855,640, filed on May 15, 2001, now Pat. No. 6,968,005.

(30) **Foreign Application Priority Data**

May 15, 2000 (GB) 0011597.2

(51) **Int. Cl.**

H04N 7/12 (2006.01)

H04N 11/02 (2006.01)

H04N 11/04 (2006.01)

H04B 1/66 (2006.01)

(52) **U.S. Cl.** **375/240.01**

(58) **Field of Classification Search** 375/240.01,
375/240.12–13, 240.15–16; 348/394.1, 401.1,
348/402.1, 403.1, 412.1, 415.1; 382/236,
382/238

See application file for complete search history.

(Continued)

OTHER PUBLICATIONS

ITU-Telecommunications Standardization Sector, Ninth Meeting:
Red Bank, NJ, Oct. 19–22, 1999.

(Continued)

Primary Examiner — Nhon Diep

(74) *Attorney, Agent, or Firm* — Alston & Bird LLP

(57) **ABSTRACT**

A method of encoding a video signal representing a sequence of pictures, the method employing both non-temporal prediction and temporal prediction, wherein the method comprises, for each picture that forms a reference picture for the temporal prediction of another picture, associating with each such picture an indicator indicating the temporal order of the reference picture in the encoded video signal relative to the other reference pictures in the encoded video signal.

62 Claims, 6 Drawing Sheets

Frame No	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Encoded Frames	I			P	P	P	P	I	P	P		P		P	P	I		P		P
RPON	1			2	3	4	5	6	7	8		9		10	11	12		13		14
TR	1			4	5	6	7	8	9	10		12		14	15	16		18		20

FOREIGN PATENT DOCUMENTS

JP	09-027944	1/1997
JP	09-098382	4/1997
JP	09-247669	9/1997
JP	10-145794	5/1998

OTHER PUBLICATIONS

ITU-Telecommunications Standardization Sector, Seventh Meeting:
Feb. 15-19, 1999, Monterey, CA, USA pp. 1-11.
“Error Concealment in Encoded Video Streams” by SALMA, et al.
pp. 1-32.
“Professor Delp’s Recent Publications” Recent Pub. of Edward J.
Delp.
“Picture Layer” 5.1 Recommendation H.263 (Feb. 1998).

“Error Control and Concealment for Video Communication: A
Review” pp. 974-997.
ITU-Telecommunications Standardization Sector, Ninth Meeting:
Red Bank, New Jersey, Oct. 19-22, 1999 p. 1.
ITU—Telecommunications Standardization Sector, Ninth Meeting:
Red bank, New Jersey , Oct. 19-22, 1999, pp. 1-3.
ITU—Telecommunications Standardization Sector, Ninth Meeting:
Red Bank, New Jersey, Oct. 19-21, 1999 p. 1-2.
ITU—Telecommunications Standardization, Sector, Nov. 1999, pp.
1-43.
Extended European Search Report for European Application No.
09010864.8 dated Jun. 28, 2010.
Office Action for Japan Application No. 2001-585523 dated Dec. 5,
2011.

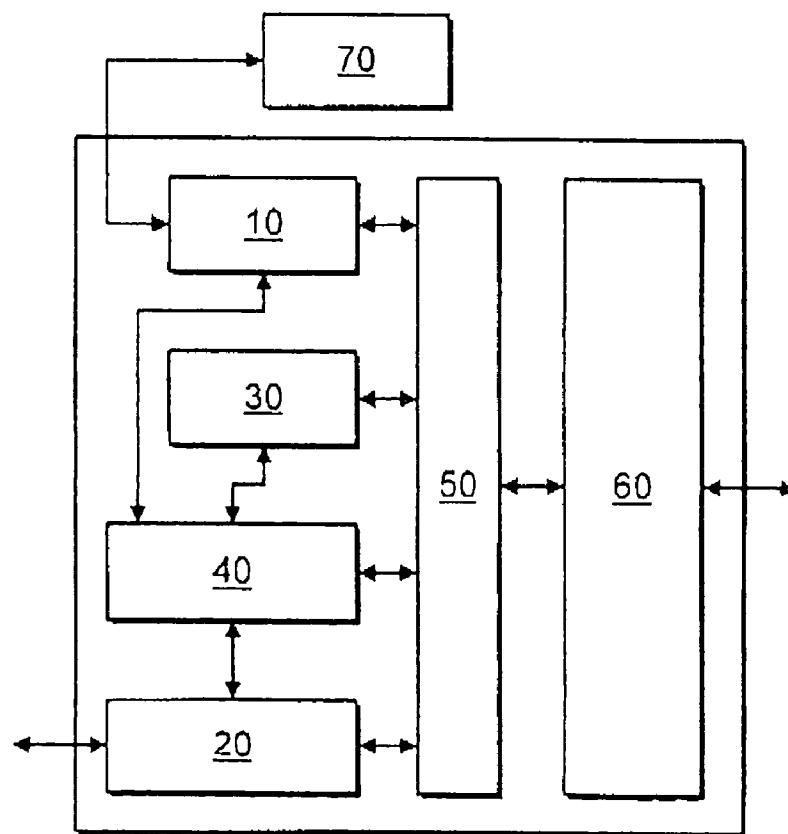
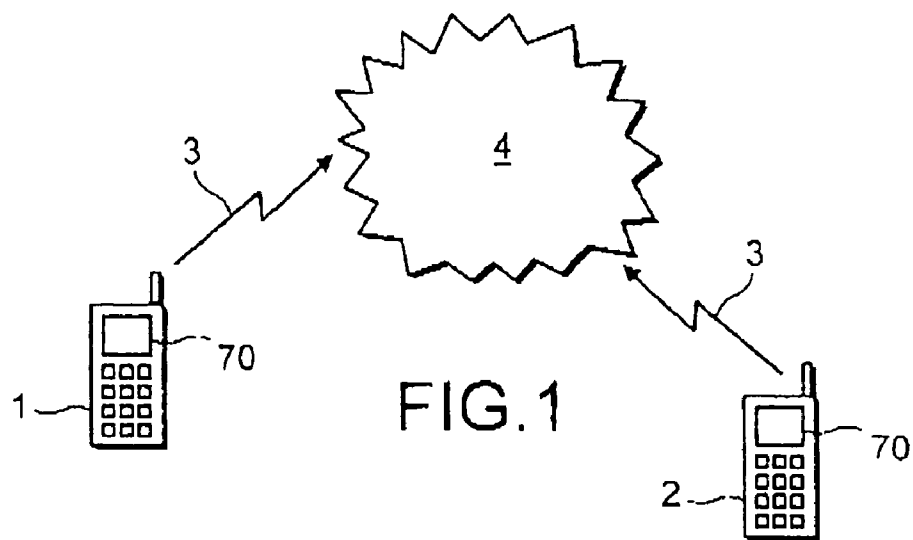


FIG. 2

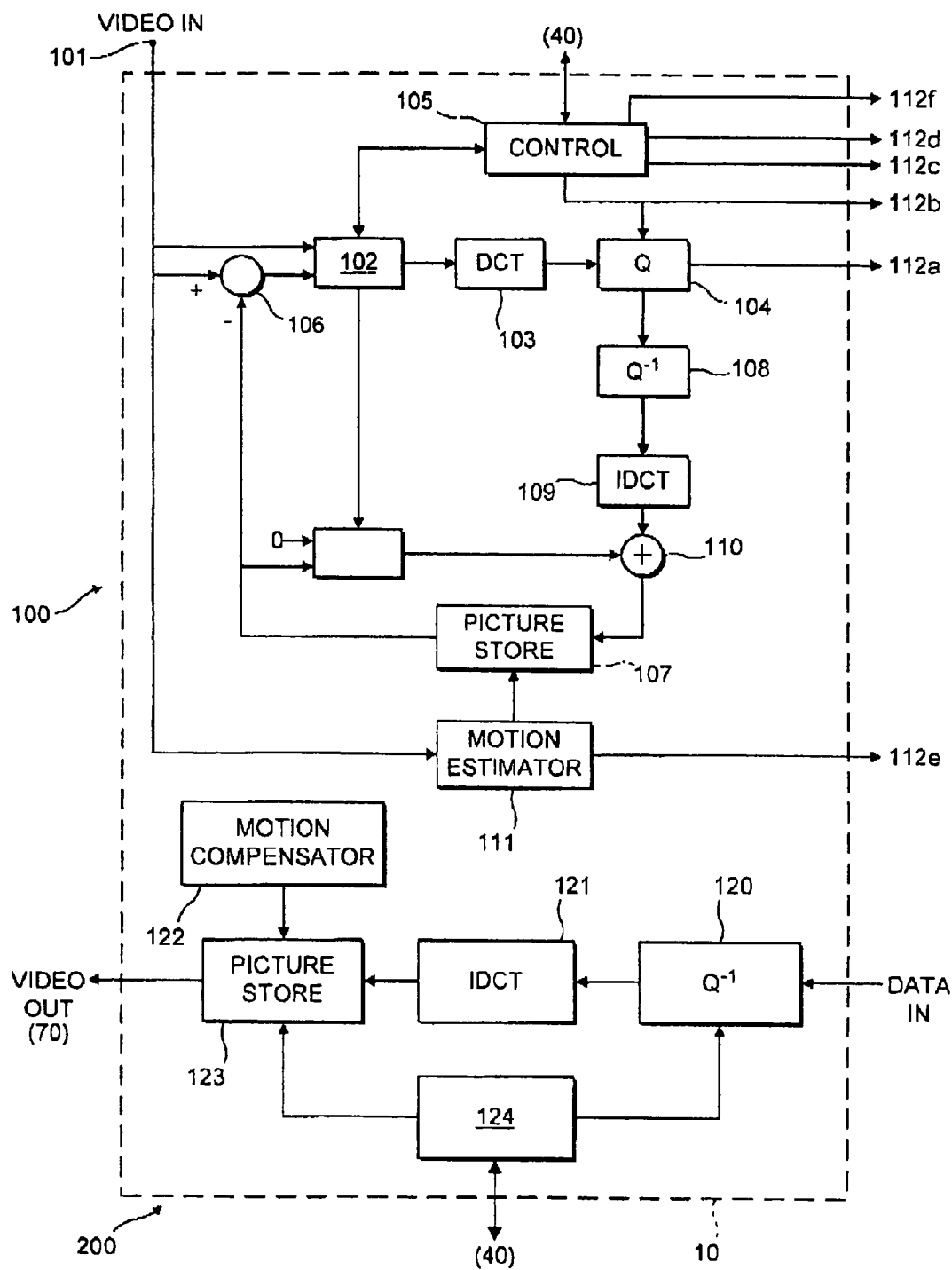


FIG.3

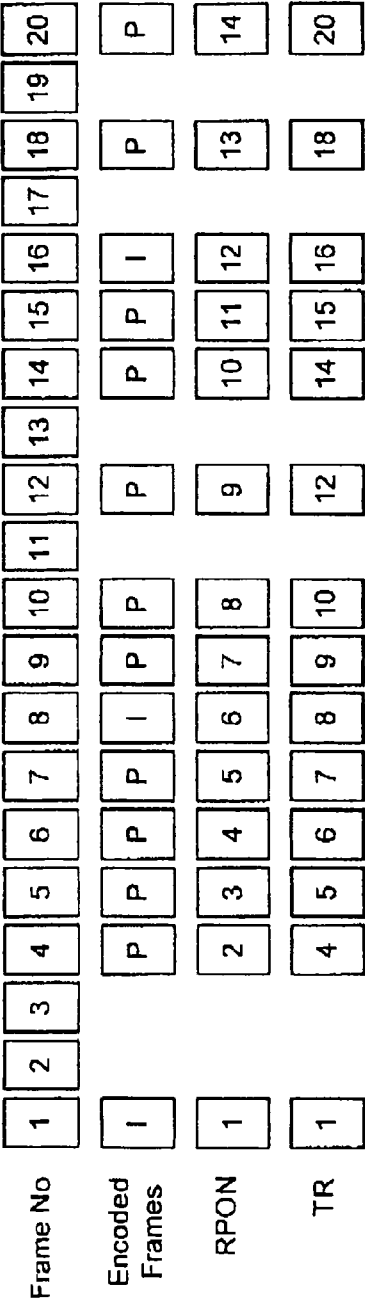


FIG. 4

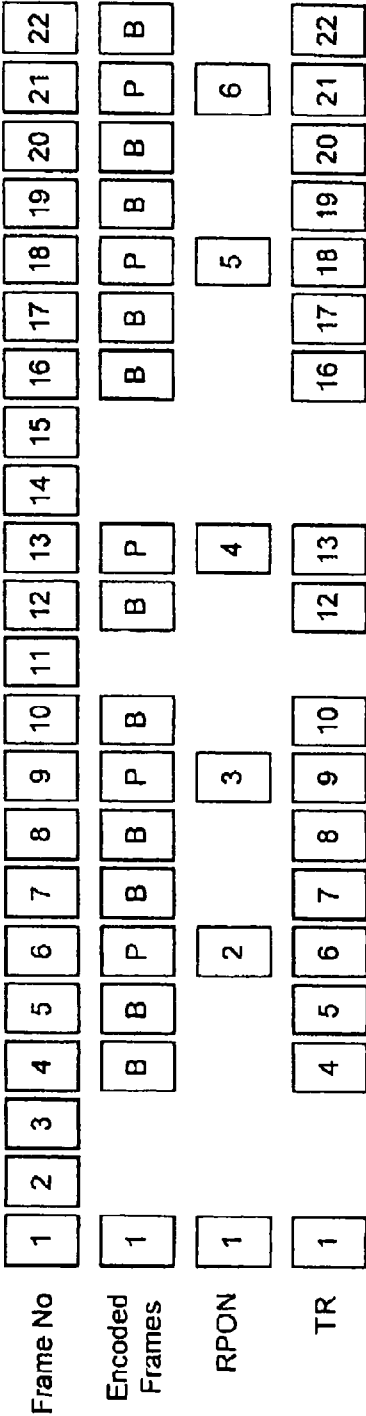


FIG. 5

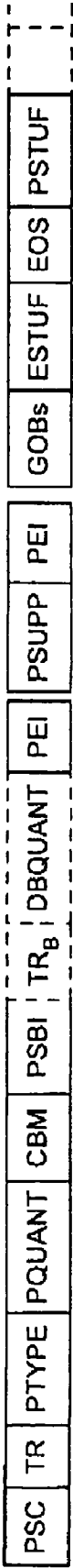
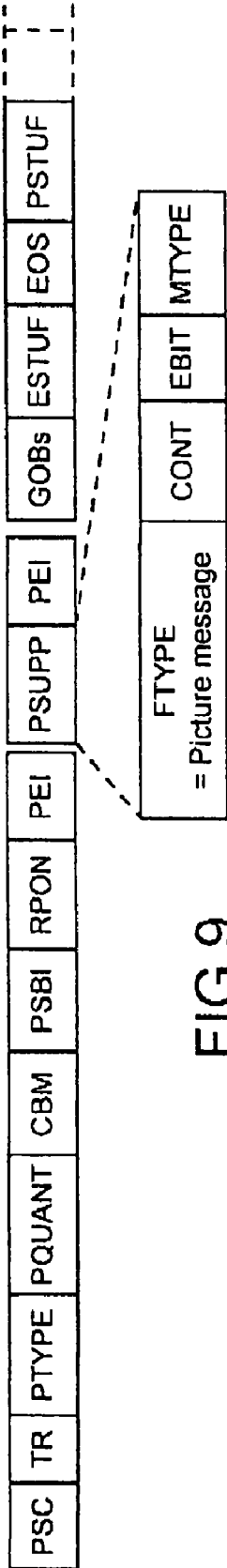
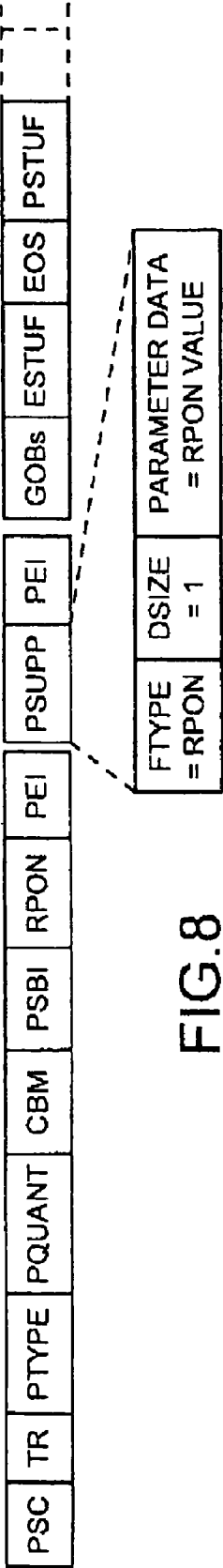


FIG. 6



FIG. 7



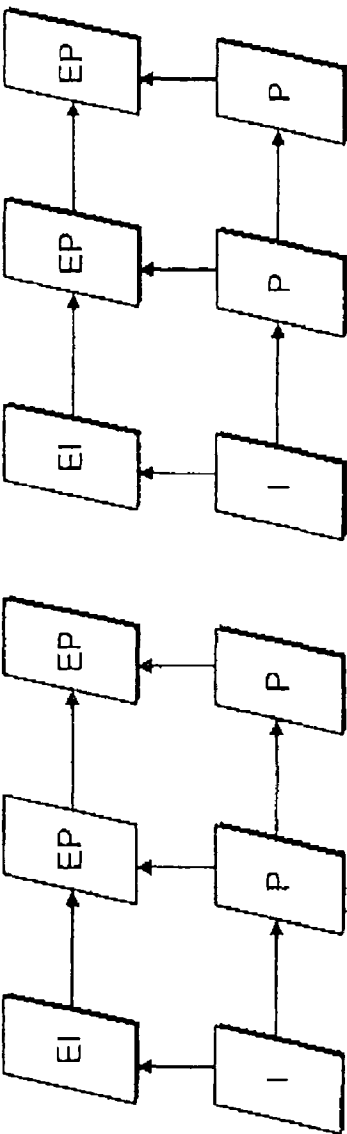


FIG. 10

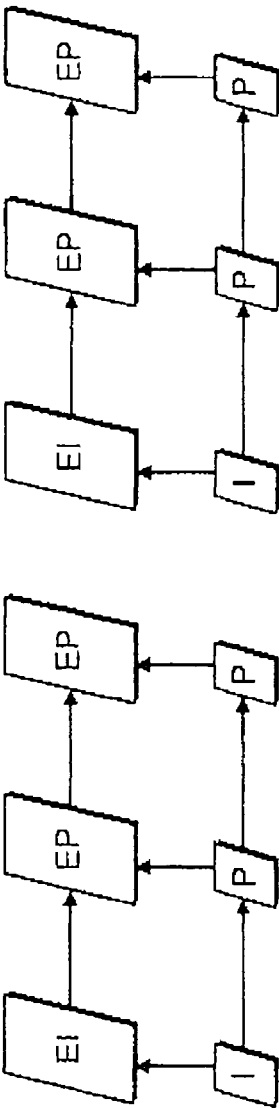


FIG. 11

1

VIDEO CODING

RELATED APPLICATIONS

This application is a continuation application of application Ser. No. 09/855,640 filed May 15, 2001 now U.S. Pat. No. 6,968,005.

BACKGROUND OF THE INVENTION

This invention related to video coding.

A video sequence consists of a series of still pictures or frames. Video compression methods are based on reducing the redundant and perceptually irrelevant parts of video sequences. The redundancy in video sequences can be categorised into spectral, spatial and temporal redundancy. Spectral redundancy refers to the similarity between the different colour components of the same picture. Spatial redundancy results from the similarity between neighbouring pixels in a picture. Temporal redundancy exists because objects appearing in a previous image are also likely to appear in the current image. Compression can be achieved by taking advantage of this temporal redundancy and predicting the current picture from another picture, termed an anchor or reference picture. Further compression is achieved by generating motion compensation data that describes the motion between the current picture and the previous picture.

However, sufficient compression cannot usually be achieved by only reducing the inherent redundancy of the sequence. Thus, video encoders also try to reduce the quality of those parts of the video sequence which are subjectively less important. In addition, the redundancy of the encoded bit-stream is reduced by means of efficient lossless coding of compression parameters and coefficients. The main technique is to use variable length codes.

Video compression methods typically differentiate between pictures that utilise temporal redundancy reduction and those that do not. Compressed pictures that do not utilise temporal redundancy reduction methods are usually called INTRA or I-frames or I-pictures. Temporally predicted images are usually forwardly predicted from a picture occurring before the current picture and are called INTER or P-frames. In the INTER frame case, the predicted motion-compensated picture is rarely precise enough and therefore a spatially compressed prediction error frame is associated with each INTER frame. INTER pictures may contain INTRA-coded areas.

Many video compression schemes also use temporally bi-directionally predicted frames, which are commonly referred to as B-pictures or B-frames. B-pictures are inserted between anchor picture pairs of I- and/or P-frames and are predicted from either one or both of these anchor pictures. B-pictures normally yield increased compression as compared with forward-predicted pictures. B-pictures are not used as anchor pictures, i.e., other pictures are not predicted from them. Therefore they can be discarded (intentionally or unintentionally) without impacting the picture quality of future pictures. Whilst B-pictures may improve compression performance as compared with P-pictures, their generation requires greater computational complexity and memory usage, and they introduce additional delays. This may not be a problem for non-real time applications such as video streaming but may cause problems in real-time applications such as video-conferencing.

A compressed video clip typically consists of a sequence of pictures, which can be roughly categorised into temporally independent INTRA pictures and temporally differentially

2

coded INTER pictures. Since the compression efficiency in INTRA pictures is normally lower than in INTER pictures, INTRA pictures are used sparingly, especially in low bit-rate applications.

A video sequence may consist of a number of scenes or shots. The picture contents may be remarkably different from one scene to another, and therefore the first picture of a scene is typically INTRA-coded. There are frequent scene changes in television and film material, whereas scene cuts are relatively rare in video conferencing. In addition, INTRA pictures are typically inserted to stop temporal propagation of transmission errors in a reconstructed video signal and to provide random access points to a video bit-stream.

Compressed video is easily corrupted by transmission errors, mainly for two reasons. Firstly, due to utilisation of temporal predictive differential coding (INTER frames), an error is propagated both spatially and temporally. In practice this means that, once an error occurs, it is easily visible to the human eye for a relatively long time. Especially susceptible are transmissions at low bit-rates where there are only a few INTRA-coded frames, so temporal error propagation is not stopped for some time. Secondly, the use of variable length codes increases the susceptibility to errors. When a bit error alters the codeword, the decoder will lose codeword synchronisation and also decode subsequent error-free codewords (comprising several bits) incorrectly until the next synchronisation (or start) code. A synchronisation code is a bit pattern which cannot be generated from any legal combination of other codewords and such codes are added to the bit stream at intervals to enable re-synchronisation. In addition, errors occur when data is lost during transmission. For example, in video applications using the unreliable UDP transport protocol in IP networks, network elements may discard parts of the encoded video bit-stream.

There are many ways for the receiver to address the corruption introduced in the transmission path. In general, on receipt of a signal, transmission errors are first detected and then corrected or concealed by the receiver. Error correction refers to the process of recovering the erroneous data perfectly as if no errors had been introduced in the first place. Error concealment refers to the process of concealing the effects of transmission errors so that they are hardly visible in the reconstructed video sequence. Typically some amount of redundancy is added by the source or transport coding in order to help error detection, correction and concealment. Error concealment techniques can be roughly classified into three categories: forward error concealment, error concealment by post-processing and interactive error concealment. The term "forward error concealment" refers to those techniques in which the transmitter side adds redundancy to the transmitted data to enhance the error resilience of the encoded data. Error concealment by post-processing refers to operations at the decoder in response to characteristics of the received signals. These methods estimate the correct representation of erroneously received data. In interactive error concealment, the transmitter and receiver co-operate in order to minimise the effect of transmission errors. These methods heavily utilise feedback information provided by the receiver. Error concealment by post-processing can also be referred to as passive error concealment whereas the other two categories represent forms of active error concealment.

There are numerous known concealment algorithms, a review of which is given by Y. Wang and Q.-F. Zhu in "Error Control and Concealment for Video Communication: A Review", Proceedings of the IEEE, Vol. 86, No. 5, May 1998, pp. 974-997 and an article by P. Salama, N. B. Shroff, and E.

J. Delp, "Error Concealment in Encoded Video," submitted to IEEE Journal on Selected Areas in Communications.

Current video coding standards define a syntax for a self-sufficient video bit-stream. The most popular standards at the time of writing are ITU-T Recommendation H.263, "Video coding for low bit rate communication", February 1998; ISO/IEC 14496-2, "Generic Coding of Audio-Visual Objects. Part 2: Visual", 1999 (known as MPEG-4); and ITU-T Recommendation H.262 (ISO/IEC 13818-2) (known as MPEG-2). These standards define a hierarchy for bit-streams and correspondingly for image sequences and images.

In H.263, the syntax has a hierarchical structure with four layers: picture, picture segment, macroblock, and block layer. The picture layer data contain parameters affecting the whole picture area and the decoding of the picture data. Most of this data is arranged in a so-called picture header.

The picture segment layer can either be a group of blocks layer or a slice layer. By default, each picture is divided into groups of blocks. A group of blocks (GOB) typically comprises 16 successive pixel lines. Data for each GOB consists of an optional GOB header followed by data for macroblocks. If the optional slice structured mode is used, each picture is divided into slices instead of GOBs. A slice contains a number of successive macroblocks in scan-order. Data for each slice consists of a slice header followed by data for the macroblocks.

Each GOB or slice is divided into macroblocks. A macroblock relates to 16x16 pixels (or 2x2 blocks) of luminance and the spatially corresponding 8x8 pixels (or block) of chrominance components. A block relates to 8x8 pixels of luminance or chrominance.

Block layer data consist of uniformly quantised discrete cosine transform coefficients, which are scanned in zigzag order, processed with a run-length encoder and coded with variable length codes. MPEG-2 and MPEG-4 layer hierarchies resemble the one in H.263.

By default these standards generally use the temporally previous reference picture (I or P) (also known as an anchor picture) as a reference for motion compensation. This piece of information is not transmitted, i.e., the bit-stream does not include information identifying the reference picture. Consequently, decoders have no means to detect if a reference picture is lost. Although many transport coders place video data into packets and associate a sequence number with the packets, these sequence numbers are not related to the video bit-stream. For example, a section of video bit-stream may contain P-picture P1, B-picture B2, P-picture P3, and P-picture P4, captured (and to be displayed) in this order. However, this section would be compressed, transmitted, and decoded in the following order: P1, P3, B2, P4 since B2 requires both P1 and P3 before it can be encoded or decoded. Assuming that there is one picture per packet, that each packet contains a sequence number and that the packet carrying B2 is lost, the receiver can detect this packet loss from the packet sequence numbers. However, the receiver has no means to detect if it has lost a motion compensation reference picture for P4 or if it has lost a B-picture, in which case it could continue decoding normally.

The decoder therefore usually sends an INTRA request to the transmitter and freezes the picture on the display. However the transmitter may not be about to respond to this request. For instance in a non-real-time video streaming application, the transmitter cannot respond to an INTRA request from a decoder. Therefore the decoder freezes the picture until the next INTRA frame is received. In a real-time application such as video-conferencing, the transmitter may not be able to respond. For instance, in a multi-party confer-

ence, the encoder may not be able to respond to individual requests. Again the decoder freezes the picture until an INTRA frame is output by the transmitter.

SUMMARY OF THE INVENTION

According to a first aspect of the invention there is provided a method of encoding a video signal representing a sequence of pictures, the method employing both non-temporal prediction and temporal prediction, wherein the method comprises, for each picture that forms a reference picture for the temporal prediction of another picture, associating with each such picture an indicator indicating the temporal order of the reference picture in the encoded video signal relative to the other reference pictures in the encoded video signal.

Thus each reference picture (e.g. I-frames and P-frames) is associated with a sequence number. Preferably the indicator is incremented each time a reference picture is encoded. Most advantageously the indicator is incremented by one each time a reference picture is encoded. Thus the indicator is incremented by one from the previous reference picture.

If multi-layer coding is used, preferably this indicator is incremented by one from the previous reference picture in the same enhancement layer.

Including this indicator means that a decoder is capable of determining whether a reference picture has been lost and to take appropriate action, if available. This is the case even if the transport protocol does not include sequence information about the packets being transmitted or the transmitter uses a varying encapsulation strategy. For example, if the transmitter encapsulates a varying number of GOBs in one packet, receivers cannot reliably know how many GOBs or entire pictures were lost even if they could detect packet losses from packet sequence numbers.

The invention also enables a decoder to differentiate B picture losses from reference picture losses. Consequently, decoders can continue decoding after a B picture loss instead of waiting for the next INTRA picture.

In addition a decoder may continue decoding lower enhancement layers if a reference picture from a higher enhancement layer is lost.

The reference picture order number may be in respect of the whole picture or part of a picture. In the former case, typically the reference picture order number is included in a picture header. In a preferred implementation of the invention, the video signal is encoded according to the H.263 standard and the indicator is included in the Supplemental Enhancement Information. Where the RPON is in respect of part of a picture the reference picture order number may be included in the picture segment headers or macroblock headers of the encoded picture.

In accordance with a second aspect of the invention there is provided a method of decoding an encoded video signal representing a sequence of pictures, the method comprising receiving an encoded video signal, decoding each received picture, examining for each picture to be decoded that forms a reference picture for another picture an indicator representing the temporal order of a reference frame and, when the indicator does not follow consecutively from an immediately preceding decoded reference frame, detecting a lost reference frame.

Preferably the decoder sends a request to a transmitter to encode a picture in a non-temporal manner when the indicator does not follow consecutively from an immediately preceding decoded reference frame.

In accordance with a third aspect of the invention there is provided an encoded video signal wherein for each picture

5

that forms a reference picture for the temporal prediction of another picture, an indicator is associated with each such reference picture indicating the temporal order of the reference pictures in the encoded video stream.

In accordance with a fourth aspect of the invention there is provided a video encoder comprising an input for receiving a video signal representing a sequence of pictures and for generating encoded pictures, said encoder being arranged to employ both non-temporal prediction and temporal prediction, wherein the encoder is arranged, for each picture that forms a reference picture for the temporal prediction of another picture, to associate with each reference picture an indicator indicating the temporal order of the reference picture in the encoded video signal relative to other reference pictures in the encoded video signal.

Preferably the indicator is incremented each time a reference picture is encoded.

In accordance with a fifth aspect of the invention there is provided a video decoder comprising an input for receiving an encoded video signal representing a sequence of pictures, a decoder for decoding each received picture, the decoder being arranged to examine for each picture to be decoded that forms a reference picture for another picture an indicator representing the temporal order of a reference frame and, when the indicator does not follow consecutively from an immediately preceding decoded reference frame, to detect a lost reference frame.

The invention also relates to a portable radio communications device including an encoder and/or a decoder as described.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will now be described, by way of example only, with reference to the accompanying drawings, in which:

FIG. 1 shows a multimedia mobile communications system;

FIG. 2 shows an example of the multimedia components of a multimedia terminal;

FIG. 3 shows an example of a video codec;

FIG. 4 illustrates the operation of a first implementation of a video encoder according to a first embodiment of the invention;

FIG. 5 illustrates the operation of a second implementation of a video encoder according to a first embodiment of the invention;

FIG. 6 shows the syntax of a bit stream as known according to H.263;

FIG. 7 shows a first example of a bit stream output by an encoder according to the invention;

FIG. 8 shows a second example of a bit stream output by an encoder according to the invention;

FIG. 9 shows a third example of a bit stream output by an encoder according to the invention;

FIG. 10 illustrates enhancement layers used in video coding for SNR scalability; and

FIG. 11 illustrates enhancement layers used in video coding for spatial scalability.

DETAILED DESCRIPTION OF THE INVENTION

FIG. 1 shows a typical multimedia mobile communications system. A first multimedia mobile terminal 1 communicates with a second multimedia mobile terminal 2 via a radio link 3 to a mobile communications network 4. Control data is sent between the two terminals 1,2 as well as the multimedia data.

6

FIG. 2 shows the typical multimedia components of a terminal 1. The terminal comprises a video codec 10, an audio codec 20, a data protocol manager 30, a control manager 40, a multiplexer/demultiplexer 50 and a modem 60 (if the required). The video codec 10 receives signals for coding from a video capture device of the terminal (not shown) (e.g. a camera) and receives signals for decoding from a remote terminal 2 for display by the terminal 1 on a display 70. The audio codec 20 receives signals for coding from the microphone (not shown) of the terminal 1 and receive signals for decoding from a remote terminal 2 for reproduction by a speaker (not shown) of the terminal 1.

The control manager 40 controls the operation of the video codec 10, the audio codec 20 and the data protocol manager 30. However, since the invention is concerned with the operation of the video codec 10, no further discussion of the audio codec 20 and protocol manager 30 will be provided.

FIG. 3 shows an example of a video codec 10 according to the invention. The video codec comprises an encoder part 100 and a decoder part 200. The encoder part 100 comprises an input 101 for receiving a video signal from a camera or video source (not shown) of the terminal 1. A switch 102 switches the encoder between an INTRA-mode of coding and an INTER-mode.

In INTRA-mode, the video signal from the input 101 is transformed into DCT co-efficients by a DCT transformer 103. The DCT coefficients are then passed to a quantiser 104 that quantises the coefficients. Both the switch 102 and the quantiser 104 are controlled by an encoding control manager 105 of the video codec, which also receives feedback control from the receiving terminal 2 by means of the control manager 40.

In INTER mode, the switch 102 is operated to accept from a subtractor 106 the difference between the signal from the input 101 and a previous picture which is stored in a picture store 107. The difference data output from the subtractor 106 represents the prediction error between the current picture and the previous picture stored in the picture store 107. The data in the picture store 107 is generated by passing the data output by the quantiser through an inverse quantiser 108 and applying an inverse DCT transform 109 to the inverse-quantised data. The resulting data is added to the contents of the picture store 107 by adder 110. A motion estimator 111 may generate motion compensation data from the data in the picture store 107 in a conventional manner.

The encoding control manager 105 decides whether to apply INTRA or INTER coding or whether to code the frame at all on the basis of either the output of the subtractor 106 or in response to feedback control data received a receiving decoder. When not responding to feedback control data, the encoder typically encodes a frame as an INTRA-frame either only at the start of coding (all other frames being P-frames), or at regular periods e.g. every 5 s, or when the output of the subtractor exceeds a threshold i.e. when the current picture and that stored in the picture store 107 are too dissimilar. The encoder may also be programmed to encode frames in a particular regular sequence e.g. I B B P B B P B B P B B P B I B B P etc. In addition the encoding control manager may decide not to code a received frame at all. This happens when the similarity between the current frame and the reference frame is so high that the encoder decides not to encode the current frame. The encoding control manager operates the switch accordingly.

The video codec outputs the quantised DCT coefficients 112a, the quantising index 112b (i.e. the details of the quantiser used), an INTRA/INTER flag 112c to indicate the mode of coding performed (I or P/B), a transmit flag 112d to indi-

cate the number of the frame being coded and the motion vectors **112e** for the picture being coded. These are multiplexed together by the multiplexer **50** together with other multimedia signals.

The decoder part **200** of the video codec **10** comprises an inverse quantiser **120**, an inverse DCT transformer **121**, a motion compensator **122**, a picture store **123** and a controller **124**. The controller **124** receives video codec control signals demultiplexed from the encoded multimedia stream by the demultiplexer **50**. In practice the controller **105** of the encoder and the controller **124** of the decoder may be the same processor.

The operation of an encoder according to the invention will now be described. The video codec **10** receives a video signal to be encoded. The encoder **100** of the video codec encodes the video signal by performing DCT transformation, quantisation and motion compensation. The decoded video data is then output to the multiplexer **50**. The multiplexer **50** multiplexes the video data from the video codec **10** and control data from the control manager **40** (as well as other signals as appropriate) into a multimedia signal. The terminal **1** outputs this multimedia signal to the receiving terminal **2** via the modem **60** (if required).

In a first embodiment of the invention, each time the encoder encodes a frame which may form the reference frame for a subsequent frame, the encoding control manager **105** associates with the frame a so-called Reference Picture Order Number (RPON). For example, a RPON is associated with every I or P frame of a video signal but not with a B-frame. The RPON value is incremented each time a successive reference picture is encoded, preferably by 1.

The encoding control manager **105** outputs the RPON codeword on output **112f** which indicates the Reference Picture Order Number associated with the encoded frame. This is multiplexed into the video bitstream by a multiplexer.

FIG. **4** illustrates the operation of the encoder. In this embodiment, the encoder is arranged to output an I-frame when the similarity between the frame being coded and the reference frame is less than a first threshold i.e. when the output from the subtractor **106** is greater than a first threshold. Otherwise the encoder outputs P-frames. The first line of FIG. **4** represents the frames of data received from a capture input device and input to the video encoder on input **101**. The second line of FIG. **4** represents those frames of the input signal that the encoder decides to encode and the coding mode used to encode each frame. As mentioned above some, the encoding control manager may decide that a frame is not to be coded; this is exemplified in FIG. **4** by the fact that frames **2**, **3** and **11** are not coded.

Frame **1** is coded in INTRA-mode; frame **4** is encoded as a P-frame with reference to frame **1**; frame **5** is encoded as a P-frame with reference to frame **4**; frame **6** is encoded as a P-frame with reference to frame **5**; frame **7** is encoded as a P-frame with reference to frame **6**; frame **8** is encoded as an I-frame; frame **9** is encoded as a P-frame with reference to frame **8**; frame **10** is encoded as a P-frame with reference to frame **9**; frame **12** is encoded as a P-frame with reference to frame **10**.

In this embodiment all (but the last) of the encoded frames act as the reference frame for a later frame. Thus a RPON is associated with all of the frames to be coded, as shown in the third line of FIG. **4**. As can be seen, the RPON is incremented by 1 each time.

The fourth line of FIG. **4** shows the Temporal Reference (TR) of the encoded frame. This is a field included in H.263 and the value of TR is formed by incrementing its value in the temporally previous reference picture header by one plus the

number of skipped or non-reference pictures since the previously transmitted one. Thus in the example shown in FIG. **4** the TR shown for each frame is the same as the original number in the original signal input to **102**.

FIG. **5** shows a second embodiment of an encoder according to the invention. In this embodiment, the encoder is arranged to code the frames according to the regular sequence I B B P B B P B B P B B P B B P. The first line of FIG. **5** shows the input frames and the second line shows the coded frames and their coding mode, I, P or B.

The frames are received from a video capture device in the order 1, 2, 3, 4, 5, 6 etc. and are displayed in this order i.e. the decoded frames are displayed in the order I1, B2, B3, P4, B5, B6, P7 etc. However the video bit stream is compressed, transmitted and decoded in the following order I1, P4, B2, B3, P7, B5, B6 etc. This is because each B-frame requires preceding and succeeding reference frames before they can be encoded/decoded i.e. frame B2 requires frame I1 and P4 to be encoded/decoded before frame B2 can be encoded/decoded.

As explained previously, B-frames are inserted between anchor picture pairs of I- and/or P-frames and are predicted from either one or both of these anchor pictures. Thus in the illustration given in FIG. **5**, Frame **1** is coded in INTRA-mode; frame **4** is encoded as a B-frame with reference to frame **1** and/or **6**; frame **5** is encoded as a B-frame with reference to frame **1** and/or **6**; frame **6** is encoded as a P-frame with reference to frame **1**; frame **7** is encoded as a B-frame with reference to frame **6** and/or **9**; frame **8** is encoded as an B-frame with reference to frame **6** and/or **9**; frame **9** is encoded as a P-frame with reference to frame **6**; frame **10** is encoded as a B-frame with reference to frame **9** and/or **13** (not shown); frame **12** is encoded as a B-frame with reference to frame **9** and/or **13** and so on.

In this embodiment each I-frame and P-frame of the encoded sequence acts as a reference frame for another frame. However a B-frame does not act as a reference picture for any other frame. Thus a RPON is associated with all of the I-frames and P-frames, as shown in the third line of FIG. **5**. As can be seen, the RPON is incremented by 1 each time. Thus frame **1** (an I-frame) has a RPON of 1, frame **4** (a P-frame) has a RPON of 2 and frame **9** (a P-frame) has a RPON of 3.

The fourth line of FIG. **5** shows the Temporal Reference (TR) of the encoded frame. As in the example shown in FIG. **4**, the TR shown for each frame is the same as the order of occurrence in the original signal input to **10**.

Considering the terminal **1** as receiving coded video data from terminal **2**, the operation of the video codec **10** will now be described with reference to its decoding role. The terminal **1** receives a multimedia signal from the transmitting terminal **2**. The demultiplexer **50** demultiplexes the multimedia signal and passes the video data to the video codec **10** and the control data to the control manager **40**. The decoder **200** of the video codec decodes the encoded video data by inverse quantising, inverse DCT transforming and motion compensating the data. The controller **124** of the decoder checks the integrity of the received data and, if an error is detected, attempts to conceal the error in a manner to be described below. The decoded, corrected and concealed video data is then output for reproduction on a display **70** of the receiving terminal **1**.

Errors in video data may occur at the picture level, the GOB level or the macroblock level. Error checking may be carried out at any or all of these levels.

Considering first the signal as shown in FIG. **4**, when a decoder according to the invention receives this signal each frame of the signal is decoded in a conventional manner and then displayed on a display means. The decoded frame may be error corrected and error concealed in a conventional man-

ner. Each time a frame is decoded, the decoder examines the TR field to determine when the frame is to be displayed. If the TRs are not consecutive (e.g. the decoder receives a frame with TR=1 and then a frame with TR=4) the decoder holds the frame **1** on the display for 3 times the usual frame period, as is conventional. The decoder also examines the RPON of the received frames. In the case shown in FIG. **4** the decoder receives frame **1** and sees that this frame has a RPON=1; the decoder then receives a frame with TR=4 and RPON=2. The decoder compares the RPON of the currently received frame with the RPON of the previously received frame and calculates the difference between the RPON values. In this case the difference is 1 and the decoder therefore knows that no reference pictures have been lost between the current frame and the previous decoded reference frame. The decoder therefore continues to decode the signal in a conventional manner.

Let us now assume that the decoder is unable to reconstruct frame **5** (this could be due to the data being greatly corrupted or being lost altogether) and the next frame received and decoded by the decoder is frame **6**. The decoder compares the RPON of the currently received frame (frame **6**) with the RPON of the previously received and decoded reference frame (frame **4**) and calculates the difference between the RPON values. In this case the difference is 2 and the decoder therefore knows that a reference picture has been lost between transmission of the current frame and that of the previous frame. If the decoder has the facility to send control feedback data to the transmitting video encoder the decoder can send a request to the transmitting video encoder to encode a frame as an INTRA-frame and so stop the temporal error propagation that would result from frame **6** being decoded with reference to frame **4**.

Considering now the signal as shown in FIG. **5**, when a decoder according to the invention receives this signal each frame of the signal is decoded in a conventional manner and then displayed on a display means. The decoded frame may be error corrected and error concealed in a conventional manner. Each time a frame is decoded, the decoder examines the TR field to determine when the frame is to be displayed. The decoder also examines the RPON of the received frames.

In the case shown in FIG. **5** the decoder receives frame **1** and sees that this frame has a RPON=1. The decoder decodes this frame in a conventional INTRA-mode manner. The next frame received by the decoder is then frame **6**, with TR=6 and RPON=2. The decoder compares the RPON of the currently received frame (frame **6**) with the RPON of the previously received and decoded reference frame (frame **1**) and calculates the difference between the RPON. In this case the difference is 1 and the decoder therefore knows that no reference pictures have been lost between transmission of the current frame and that of the previous decoded reference frame. The decoder then decodes frame **6** with reference to frame **1**.

The decoder then receives a frame with TR=4 and no RPON. In this case the decoder makes no further use of the RPON and decodes frame **4** with reference to decoded frames **1** and **6**.

Let us now assume that the decoder is unable to reconstruct frame **5** (this could be due to the data being greatly corrupted or being lost altogether). The fact that B-frame **5** has been lost is of no consequence to the decoder as the B-frame does not form a reference picture for any other frame and thus its loss will not introduce any temporal error propagation.

The next frame to be received is frame **9**. However, let us now assume that the decoder is unable to reconstruct frame **9**, which is a P-frame (this could be due to the data being greatly corrupted or being lost altogether). The decoder may therefore be unable to decode successfully any of frames **7**, **8**, **10** or

12 since these may all be predicted, in part at least, with reference to frame **9**. Typically, in this situation, the decoder will freeze the displayed picture.

The next frame received and decoded by the decoder is frame **13**. The decoder compares the RPON of the currently received reference frame (frame **13**) with the RPON of the previously received and decoded reference frame (frame **6**) and calculates the difference between the RPON. In this case the difference is 2 and the decoder therefore knows that a reference picture has been lost between the current frame and the previous decoded reference frame. If the decoder has the facility to send control feedback data to the transmitting video encoder the decoder can send a request to the transmitting video encoder to encode a frame as an INTRA-frame and so stop the temporal error propagation that would result from frame **13** being decoded with reference to frame **6**.

How the reference picture order number may be included in the encoded signal will now be addressed with reference to the H.263 video coding standard.

FIG. **6** shows the syntax of a bit stream as known according to H.263. The following implementations describe the GOB format but it will be clear to a skilled person that the invention may also be implemented in the slice format.

As mentioned already, the bit stream has four layers: the picture layer, picture segment layer, macroblock layer and block layer. The picture layer comprises a picture header followed by data for the Group of Blocks, eventually followed by any optional end-of-sequence code and stuffing bits.

The prior art H.263 bit stream is formatted as shown in FIG. **6**. A descriptor for each part is given below:

PSC The picture start code (PSC) indicates the start of the picture
 TR The Temporal Reference (TR) is formed by incrementing its value in the temporally previous reference picture header by one plus the number of skipped or non-referenced pictures since the previously transmitted one
 PTYPE Amongst other things, PTYPE includes details of the picture coding type i.e. INTRA or INTER
 PQQUANT A codeword that indicates the quantiser to be used for the picture until updated by any subsequent quantiser information
 CPM A codeword that signals the use of optional continuous presence multipoint and video multiplex (CPM) mode
 PSBI Picture Sub-Bit stream Indicator—only present if CPM is set
 TR_B Present if the frame is a bi-directionally predicted frame (known as a PB-frame)
 DBQUANT Present if a bidirectional frame
 PEI This relates to extra insertion information and is set to “1” to indicate the presence of the following optional data fields PSUPP and PEI. PSUPP and PEI are together known as Supplemental Enhancement Information, which is further defined in Annex L of H263.
 GOBS Is the data for the group of blocks for the current picture
 ESTF A stuffing codeword provided to attain byte alignment before EOS
 EOS A codeword indicating the end of the data sequence for the picture
 PSTUF A stuffing codeword to allow for byte alignment of the next picture start code PSC

The structure as shown in FIG. **4** does not include the optional PLUSTYPE data field. PSBI is only present if indicated by CPM. TRB and DBQUANT are only present if PTYPE indicates use of a so-called PB frame mode (unless

the PLUSTYPE field is present and the use of DBQUANT is indicated therein). These issues are addressed in more detail in the H.263 specification.

The following paragraphs outline possible implementations of the bit-stream output by an encoder according to the first aspect of the invention.

The reference picture order number may be incorporated into a H.263 bit stream as follows. FIG. 7 shows an example of a bit stream output by an encoder according to the first implementation of the invention. As shown in FIG. 7, the bit stream includes a further codeword RPON which is a codeword indicating the reference picture order number. This is inserted by an encoder according to the invention, as described above.

Alternatively, the reference picture order number may be included in the Supplemental Enhancement Information PSUPP (see Annex L of H.263 and FIG. 4). The supplemental information may be present in the bit stream even though the decoder may not be capable of providing the enhanced capability to use it, or even to properly interpret it. Simply discarding the supplemental information is allowable by decoders unless a requirement to provide the requested capability has been negotiated by the transmitter and receiver.

If PEI is set to "1", then 9 bits follow consisting of 8 bits of data (PSUPP) and then another PEI bit to indicate if a further 9 bits follow and so on.

The PSUPP data consists of a 4-bit function type indication FTYPE, followed by a 4-bit parameter data size specification DSIZE followed by DSIZE octets of function parameter data, optionally followed by another FTYPE and so on. It is known to use this PSUPP codeword to signal various situations such as: to indicate a full-picture or partial-picture freeze or freeze-release request with or without resizing; to tag particular pictures or sequences of pictures within the video stream for external use; or to convey chroma key information for video compositing.

To implement the invention using the Supplemental Enhancement Information, a further FTYPE is defined as Reference Picture Order Number.

FIG. 8 illustrates the example where a parameter RPON is included in the SEI of the picture header. The FTYPE is defined as RPON. The DSIZE specifies the size of the parameter and the following octet is the parameter data i.e. the value of RPON. From this value a receiving decoder can determine whether a reference picture has been lost.

Alternatively, the information may be contained in the additional Supplemental Enhancement Information as specified in a "Draft of new Annex W: Additional Supplementary Enhancement Information Specification" P. Ning and S. Wenger, ITU-T Study Group 16 Question 15 Document Q15-I-58, November 1999.

In this draft proposal, FTYPE 14 is defined as "Picture Message". When this FTYPE is set, the picture message function indicates the presence of one or more octets representing message data. The first octet of the message data is a message header with the structure shown in FIG. 9 i.e. CONT, EBIT and MTYPE. DSIZE is equal to the number of octets in the message data corresponding to a picture message function, including the first octet message header.

The continuation field CONT, if equal to 1, indicates that the message data associated with the picture message is part of the same logical message as the message data associated with the next picture message function. The End Bit Position field EBIT specifies the number of least significant bits that shall be ignored in the last message octet. Further details of these fields can be found in the draft of Annex W referred to above.

The field MTYPE indicates the type of message. Various types of message are suggested in the draft of Annex W. According to the invention one type e.g. MTYPE 12 is defined as RPON or Picture Number. The message contains two data bytes that carry a 10-bit Picture Number. Consequently, DSIZE shall be 3, CONT shall be 0, and EBIT shall be 6. Picture Number shall be incremented by 1 for each coded and transmitted I or P picture or PB or Improved PB frame, in a 10-bit modulo operation. For EI and EP pictures, Picture Number shall be incremented for each EI or EP picture within the same scalability enhancement layer. For B pictures, Picture Number shall be incremented relative to the value in the most recent non-B picture in the reference layer of the B picture which precedes the B picture in bitstream order (a picture which is temporally subsequent to the B picture). If adjacent pictures in the same enhancement layer have the same temporal reference, and if the reference picture selection mode (see Annex N) is in use, the decoder shall regard this occurrence as an indication that redundant copies have been sent of approximately the same pictured scene content, and all of these pictures shall share the same Picture Number. If the difference (modulo 1024) of the Picture Numbers of two consecutively received non-B pictures in the same enhancement layer is not 1, and if the pictures do not represent approximately the same pictured scene content as described above, a loss of pictures or corruption of data may be inferred by the decoder. The value of RPON is defined in the octet following the message header.

In a specific example, this message contains one data byte, i.e., DSIZE is 2, CONT is 0, and EBIT is 0.

The Reference Picture Order Number is incremented by one from the corresponding number of the previous coded reference picture. The least significant 8-bits of the result of the incrementation is placed in the data byte associated with this message.

The invention may also be implemented in accordance with Annex U of H.263.

The above description has made reference to encoded video streams in which bi-directionally predicted pictures (B-pictures) are encoded. As mentioned earlier, B-pictures are never used as reference pictures. Since they can be discarded without impacting the picture quality of future pictures, they provide temporal scalability. Scalability allows for the decoding of a compressed video sequence at more than one quality level. In other words, a scalable multimedia clip can be compressed so that it can be streamed over channels with different data rates and still be decoded and played back in real-time.

Thus the video stream may be decoded in different ways by differing decoders. For instance, a decoder can decide only to decode the I- and P-pictures of a signal, if this is the maximum rate of decoding that the decoder can attain. However if a decoder has the capacity, it can also decode the B-pictures and hence increase the picture display rate. Thus the perceived picture quality of the displayed picture will be enhanced over a decoder that only decodes the I- and P-pictures.

Scalable multimedia is typically ordered so that there are hierarchical layers of data. A base layer contains a basic representation of the multimedia clip whereas enhancement layers contain refinement data on top of underlying layers. Consequently, the enhancement layers improve the quality of the clip.

Scalability is a desirable property for heterogeneous and error prone environments. This property is desirable in order to counter limitations such as constraints on bit rate, display resolution, network throughput, and decoder complexity.

Scalability can be used to improve error resilience in a transport system where layered coding is combined with transport prioritisation. The term transport prioritisation here refers to various mechanisms to provide different qualities of service in transport, including unequal error protection, to provide different channels having different error/loss rates. Depending on their nature, data are assigned differently. For example, the base layer may be delivered through a channel with a high degree of error protection, and the enhancement layers may be transmitted through more error-prone channels.

Generally, scalable multimedia coding suffers from a worse compression efficiency than non-scalable coding. In other words, a multimedia clip encoded as a scalable multimedia clip with enhancement layers requires greater bandwidth than if it had been coded as a non-scalable single-layer clip with equal quality. However, exceptions to this general rule exist, for example the temporally scalable B-frames in video compression.

The invention may be applied to other scalable video compression systems. For instance, in H.263 Annex O, two other forms of scalability are defined: signal-to-noise (SNR) scalability and spatial scalability. Spatial scalability and SNR scalability are closely related, the only difference being the increased spatial resolution provided by spatial scalability. An example of SNR scalable pictures is shown in FIG. 10. SNR scalability implies the creation of multi-rate bit streams. It allows for the recovery of coding errors, or differences between an original picture and its reconstruction. This is achieved by using a finer quantiser to encode the difference picture in an enhancement layer. This additional information increases the SNR of the overall reproduced picture.

Spatial scalability allows for the creation of multi-resolution bit streams to meet varying display requirements and/or constraints. A spatially scalable structure is illustrated in FIG. 11. It is essentially the same as in SNR scalability except that a spatial enhancement layer attempts to recover the coding loss between an up-sampled version of the reconstructed reference layer picture and a higher resolution version of the original picture. For example, if the reference layer has a quarter common intermediate format (QCIF) resolution, and the enhancement layer has a common intermediate format (CIF) resolution, the reference layer picture must be scaled accordingly such that the enhancement layer picture can be predicted from it. The QCIF standard allows the resolution to be increased by a factor of two in the vertical direction only, horizontal direction only, or both the vertical and horizontal directions for a single enhancement layer. There can be multiple enhancement layers, each increasing the picture resolution over that of the previous layer. The interpolation filters used to up-sample the reference layer picture are explicitly defined in the H.263 standard. Aside from the up-sampling process from the reference to the enhancement layer, the processing and syntax of a spatially scaled picture are identical to those of an SNR scaled picture.

In either SNR or spatial scalability, the enhancement layer pictures are referred to as EI- or EP-pictures. If the enhancement layer picture is upwardly predicted from a picture in the reference layer, then the enhancement layer picture is referred to as an Enhancement-I (EI) picture. In this type of scalability, the reference layer means the layer "below" the current enhancement layer. In some cases, when reference layer pictures are poorly predicted, over-coding of static parts of the picture can occur in the enhancement layer, causing an unnecessarily excessive bit rate. To avoid this problem, forward prediction is permitted in the enhancement layer. A picture that can be predicted in the forward direction from a previous

enhancement layer picture or, alternatively, upwardly predicted from the reference layer picture is referred to as an Enhancement-P (EP) picture. Note that computing the average of the upwardly and forwardly predicted pictures can provide bi-directional prediction for EP-pictures. For both EI- and EP-pictures, upward prediction from the reference layer picture implies that no motion vectors are required. In the case of forward prediction for EP-pictures, motion vectors are required.

According to the invention, if the encoder is capable of multi-layer coding (for example as discussed in Annex O of H.263) the reference pictures of each layer are given consecutive Reference Picture Order Numbers. These may be associated with the enhancement layer number (ELNUM) of the current picture. The Reference Picture Order Number is incremented by one from the corresponding number of the previous coded reference picture in the same enhancement layer.

As shown in FIGS. 10 and 11, the pictures of the enhancement layer may be predicted from a preceding picture of the enhancement layer and/or from the equivalent I- or P-picture of the base layer. The enhancement layer may not be predicted from a B-picture in the reference layer.

If adjacent pictures in the same enhancement layer have the same temporal reference, and if Annex N or Annex U of H.263 is in use, the decoder preferably regards this occurrence as an indication that redundant copies have been sent of approximately the same pictured scene content, and all of these pictures then share the same RPON.

A decoder according to the invention, on receipt of a multi-layer signal as described above, attempts to decode the signal in a conventional manner. In each layer, each time a reference picture is decoded, the decoder examines the RPON of the decoded picture. If the decoder determines that a reference picture has been lost from an enhancement layer, the decoder ceases to display pictures from the enhancement layer until an EI-picture is received. The decoder continues to decode the base layer as described earlier.

The invention is not intended to be limited to the video coding protocols discussed above: these are intended to be merely exemplary. The invention is applicable to any video coding protocol in which temporal prediction may be used. The addition of the information as discussed above allows a receiving decoder to determine that a reference picture has been lost and to take appropriate action.

What is claimed is:

1. A method of encoding a video signal using an encoder to form an encoded video signal, the video signal representing a sequence of pictures, the method comprising the encoder using an independent numbering scheme, to assign consecutive reference pictures in encoding order with respective sequence indicator values that differ with respect to each other by a predetermined amount independent of one or more of the number of non-reference pictures encoded between consecutive reference pictures and the number of non-coded pictures between consecutive reference pictures.

2. A method according to claim 1, comprising the encoder incrementing the sequence indicator value by one between consecutively encoded reference pictures.

3. A method according to claim 1, comprising the encoder providing the sequence indicator value for a particular picture in a corresponding picture header.

4. A method according to claim 1, comprising the encoder encoding the video signal according to the H.263 video coding standard and providing the sequence indicator value for a particular picture in the corresponding Supplemental Enhancement Information of the H.263 bit-stream.

15

5. A method according to claim 1, comprising the encoder associating a sequence indicator value with the whole of a picture.

6. A method according to claim 1, comprising the encoder associating a sequence indicator value with part of a picture.

7. A method according to claim 6, comprising the encoder providing the sequence indicator value in a picture segment header or a macroblock header of an encoded picture.

8. A method according to claim 1, comprising the encoder encoding the video signal using multi-layer coding to produce an encoded video signal with multiple layers and providing respective sequence indicator values for each of said multiple layers of the multi-layer coded video signal.

9. A method according to claim 1, comprising the encoder using a fixed value as the predetermined amount.

10. A method according to claim 1, comprising the encoder using sequence indicator values which are reference picture order numbers.

11. A method according to claim 1, comprising the encoder encoding reference pictures in I-picture format or P-picture format.

12. A method according to claim 1, comprising the encoder encoding non-reference pictures in B-picture format.

13. A method according to claim 1, comprising the encoder incrementing the sequence indicator value by said predetermined amount between consecutively encoded reference pictures when there are no non-reference pictures.

14. A non-transitory computer readable storage medium comprising a computer program embodied thereon, the computer program comprising computer program code for performing the encoding method of claim 1.

15. A method of decoding an encoded video signal using a decoder to form a decoded video signal, the video signal representing a sequence of pictures, the method comprising:

the decoder examining decoded reference pictures to identify a difference in respective sequence indicator values assigned to consecutively encoded reference pictures; the decoder comparing the identified difference in said sequence indicator values on the basis of an independent numbering scheme in which consecutive reference pictures in encoding order are assigned sequence indicator values that differ with respect to each other by a predetermined amount, independent of one or more of the number of non-reference pictures encoded between consecutive reference pictures, and the number of non-coded pictures between consecutive reference pictures; and

the decoder detecting corruption or loss of a reference picture if said identified difference in sequence indicator values is more than said predetermined amount.

16. A method according to claim 15, comprising the decoder comparing the sequence indicator value assigned to a particular received and decoded reference picture with the sequence indicator value assigned to an immediately preceding received and decoded reference picture.

17. A method according to claim 15, further comprising the decoder sending a request for an encoder to provide a picture encoded in a non-temporally-predicted manner when said identified difference differs from said predetermined amount.

18. A method according to claim 15, comprising the decoder using a predetermined amount of one in said comparison when the sequence indicator values assigned to consecutive reference pictures in encoding order differ by one.

19. A method according to claim 15, comprising the decoder obtaining the sequence indicator value for a particular picture from a corresponding picture header.

16

20. A method according to claim 15, comprising the decoder receiving a video signal encoded according to the H.263 video coding standard and obtaining the sequence indicator value for a particular picture from the corresponding Supplemental Enhancement Information of the H.263 bit-stream.

21. A method according to claim 15, wherein a sequence indicator value is associated with the whole of a picture.

22. A method according to claim 15, wherein a sequence indicator value is associated with part of a picture.

23. A method according to claim 22, comprising the decoder obtaining the sequence indicator value from a picture segment header or a macroblock header of an encoded picture.

24. A method according to claim 15, comprising the decoder examining identified differences in sequence indicator values provided for each layer of an encoded video signal that comprises multiple layers in order to detect corruption or loss of reference pictures within said layers.

25. A method according to claim 15, wherein the predetermined amount is a fixed value.

26. A method according to claim 15, wherein the sequence indicator value is a reference picture order number.

27. A method according to claim 15, wherein reference pictures are encoded in I-picture format or P-picture format.

28. A method according to claim 15, wherein non-reference pictures are encoded in B-picture format.

29. A method according to claim 15, wherein the sequence indicator value is incremented by said predetermined amount between consecutively encoded reference pictures when there are no non-reference pictures.

30. A non-transitory computer readable storage medium comprising a computer program embodied thereon, the computer program comprising computer program code for performing the decoding method of claim 15.

31. An apparatus for encoding a video signal representing a sequence of pictures to form an encoded video signal, wherein the apparatus is configured to use an independent numbering scheme to assign consecutive reference pictures in encoding order with respective sequence indicator values that differ with respect to each other by a predetermined amount independent of one or more of the number of non-reference pictures encoded between consecutive reference pictures, and the number of non-coded pictures between consecutive reference pictures.

32. An apparatus according to claim 31, wherein the apparatus is configured to increment the sequence indicator value by one between consecutively encoded reference pictures.

33. An apparatus according to claim 31, wherein the apparatus is configured to provide the sequence indicator value for a particular picture in a corresponding picture header.

34. An apparatus according to claim 31, wherein the apparatus is configured to encode the video signal according to the H.263 video coding standard and to provide the sequence indicator value for a particular picture in the corresponding Supplemental Enhancement Information of the H.263 bit-stream.

35. An apparatus according to claim 31, wherein the apparatus is configured to associate a sequence indicator value with the whole of a picture.

36. An apparatus according to claim 31, wherein the apparatus is configured to associate a sequence indicator value with part of a picture.

37. An apparatus according to claim 36, wherein the apparatus is configured to provide the sequence indicator value in a picture segment header or a macroblock header of an encoded picture.

17

38. An apparatus according to claim 31, wherein the apparatus is configured to encode the video signal using multi-layer coding to produce an encoded video signal with multiple layers and to provide respective sequence indicator values for each of said multiple layers of the multi-layer coded video signal.

39. A portable radio communications device comprising an apparatus for encoding a video signal according to claim 31.

40. A multimedia terminal device comprising an apparatus for encoding a video signal according to claim 31.

41. An apparatus according to claim 31, wherein the apparatus is configured to use a fixed value as the predetermined amount.

42. An apparatus according to claim 31, wherein the apparatus is configured to use sequence indicator values which are reference picture order numbers.

43. An apparatus according to claim 31, wherein the apparatus is configured to encode reference pictures in I-picture format or P-picture format.

44. An apparatus according to claim 31, wherein the apparatus is configured to encode non-reference pictures in B-picture format.

45. An apparatus according to claim 31, wherein the apparatus is configured to increment the sequence indicator value by said predetermined amount between consecutively encoded reference pictures when there are no non-reference pictures.

46. An apparatus for decoding an encoded video signal representing a sequence of pictures to form a decoded video signal, wherein the apparatus is configured to:

examine decoded reference pictures to identify a difference in respective sequence indicator values assigned to consecutively encoded reference pictures;

compare the identified difference in sequence indicator values on the basis of an independent numbering scheme in which consecutive reference pictures in encoding order are assigned sequence indicator values that differ with respect to each other by a predetermined amount, independent of one or more of the number of non-reference pictures encoded between consecutive reference pictures, and the number of non-coded pictures between consecutive reference pictures; and

detect corruption or loss of a reference picture if said identified difference in sequence indicator values is more than said predetermined amount.

47. An apparatus according to claim 46, wherein the apparatus is configured to compare the sequence indicator value assigned to a particular received and decoded reference picture with the sequence indicator value assigned to an immediately preceding received and decoded reference picture.

48. An apparatus according to claim 46, wherein the apparatus is further configured to send a request for an encoder to

18

provide a picture encoded in a non-temporally-predicted manner when said identified difference differs from said predetermined amount.

49. An apparatus according to claim 46, wherein the apparatus is configured to use a predetermined amount of one in said comparison when the sequence indicator values assigned to consecutive reference pictures in encoding order differ by one.

50. An apparatus according to claim 46, wherein the apparatus is configured to obtain the sequence indicator value for a particular picture from a corresponding picture header.

51. An apparatus according to claim 46, wherein the apparatus is configured to receive a video signal encoded according to the H.263 video coding standard and to obtain a sequence indicator value for a particular picture from the corresponding Supplemental Enhancement Information of the H.263 bit-stream.

52. An apparatus according to claim 46, wherein a sequence indicator value is associated with the whole of a picture.

53. An apparatus according to claim 46, wherein a sequence indicator value is associated with part of a picture.

54. An apparatus according to claim 53, wherein the apparatus is configured to obtain the sequence indicator value from a picture segment header or a macroblock header of an encoded picture.

55. An apparatus according to claim 46, wherein the apparatus is configured to examine identified differences in sequence indicator values provided for each layer of an encoded video signal that comprises multiple layers in order to detect corruption or loss of reference pictures within said layers.

56. A portable radio communications device comprising an apparatus for decoding an encoded video signal according to claim 46.

57. A multimedia terminal device comprising an apparatus for decoding an encoded video signal according to claim 46.

58. An apparatus according to claim 46, wherein the predetermined amount is a fixed value.

59. An apparatus according to claim 46, wherein the sequence indicator value is a reference picture order number.

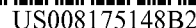
60. An apparatus according to claim 46, wherein the apparatus is configured to decode reference pictures encoded in I-picture format or P-picture format.

61. An apparatus according to claim 46, wherein the apparatus is configured to decode non-reference pictures encoded in B-picture format.

62. An apparatus according to claim 46, wherein the sequence indicator value is incremented by said predetermined amount between consecutively encoded reference pictures when there are no non-reference pictures.

* * * * *

EXHIBIT 14



(10) **Patent No.:** US 8,175,148 B2
(45) **Date of Patent:** *May 8, 2012

- | | | | | |
|--------------|------|---------|-----------------------|------------|
| 5,237,410 | A | 8/1993 | Inoue | 358/136 |
| 5,715,009 | A | 2/1998 | Tahara et al. | 348/423 |
| 5,751,358 | A | 5/1998 | Suzuki et al. | 348/405 |
| 5,835,030 | A | 11/1998 | Tsutsui et al. | 341/51 |
| 5,907,362 | A | 5/1999 | Yamamoto | 348/405 |
| 6,256,349 | B1 | 7/2001 | Suzuki et al. | 375/240.18 |
| 6,263,022 | B1 | 7/2001 | Chen et al. | 375/240 |
| 6,677,868 | B2 * | 1/2004 | Kerofsky et al. | 341/107 |
| 6,879,632 | B1 | 4/2005 | Yokoyama | 375/240 |
| 7,263,125 | B2 * | 8/2007 | Lainema | 375/240.04 |
| 7,367,041 | B2 | 4/2008 | Morishita et al. | 725/37 |
| 2003/0152146 | A1 * | 8/2003 | Lin et al. | 375/240.16 |

FOREIGN PATENT DOCUMENTS

JP	06-121171	4/1994
JP	11-004449	1/1999
WO	WO 00 21302 A1	4/2000
WO	WO 01/26381 A1	4/2001

OTHER PUBLICATIONS

Single-pass constant-and variable-bit-rate MPEG-2 video compression, N. Mohsenian, R. Rajagopalan, C.A. Gonzales, Jul. 4, 1999, XP-002216512.

(Continued)

Primary Examiner — Gims Philippe

(57) **ABSTRACT**

US 2007/0291849 A1 Dec. 20, 2007

Related U.S. Application Data

- A method and device for coding of digital video sequence, wherein an indication of quantization parameter (QP) is provided in the encoded bit-stream for decoding purposes. The QP related information is indicated by introducing a sequence level quantization parameter value SQP. More specifically, instead of coding the absolute values of picture/slice QPs, an indication of the difference ΔQP between the sequence level quantization parameter SQP and the picture/slice QP is provided. This eliminates the need to transmit a full QP for every picture/slice, and enables a statistically smaller difference value to be transmitted, thus providing a reduction in transmission bit-rate. The difference value is subsequently used in a corresponding decoder to reconstruct the picture/slice QP.

- 23 Claims, 8 Drawing Sheets**

(52) **U.S. Cl.** 375/240.03; 375/240.16

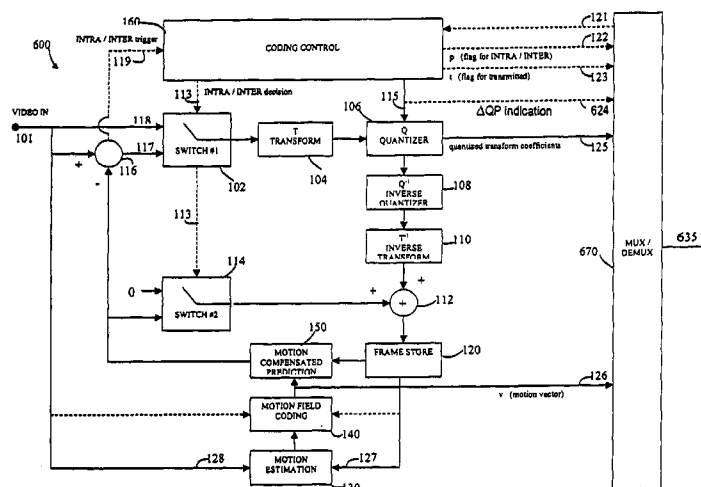
(58) **Field of Classification Search** 375/240.04,
375/240.16; 341/107

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,054,103	A	10/1991	Yasuda et al.	382/56
5,144,426	A	9/1992	Tanaka et al.	358/133
5,231,484	A	7/1993	Gonzales et al.	358/133



OTHER PUBLICATIONS

A Video Compression Algorithm With Adaptive Bit Allocation and Quantization, Eric Viscito and Cesar Gonzales, Nov. 11, 1991, Visual Communications and Image Processing '91: Visual Communication, Boston—SPIE vol. 1605.

“Joint Model No. 1 (JM-1)”, Doc. JVT-A003, Appendix B; Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG; Jan. 2002; pp. 1-79.

* cited by examiner

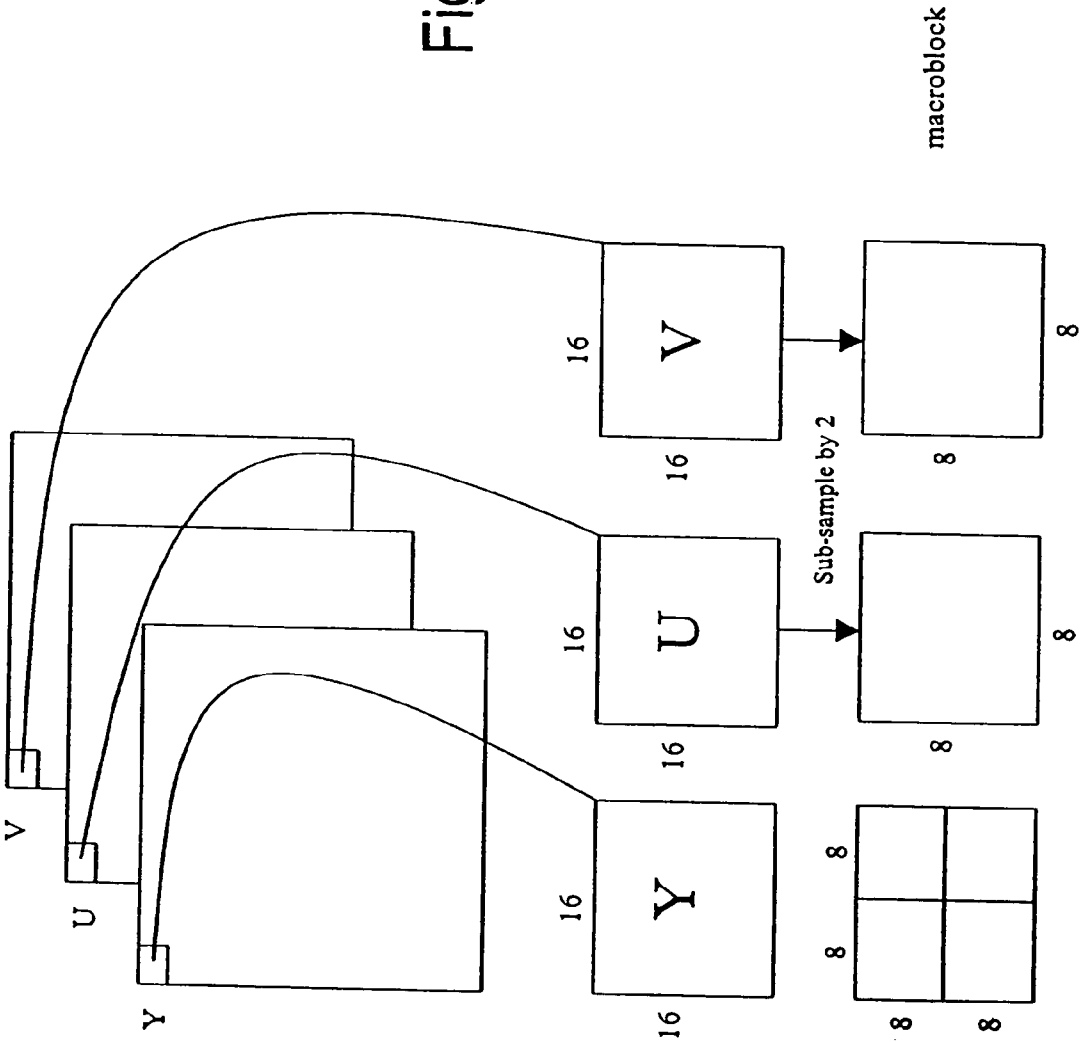
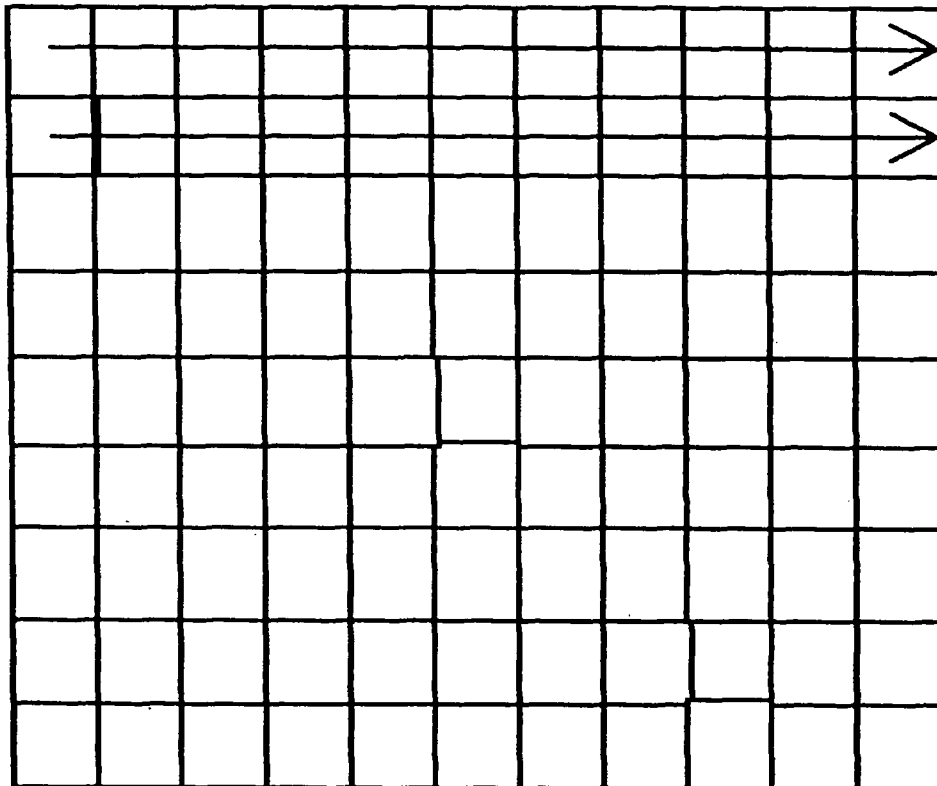


Figure 1

QCIF Image



9

11

Figure 2

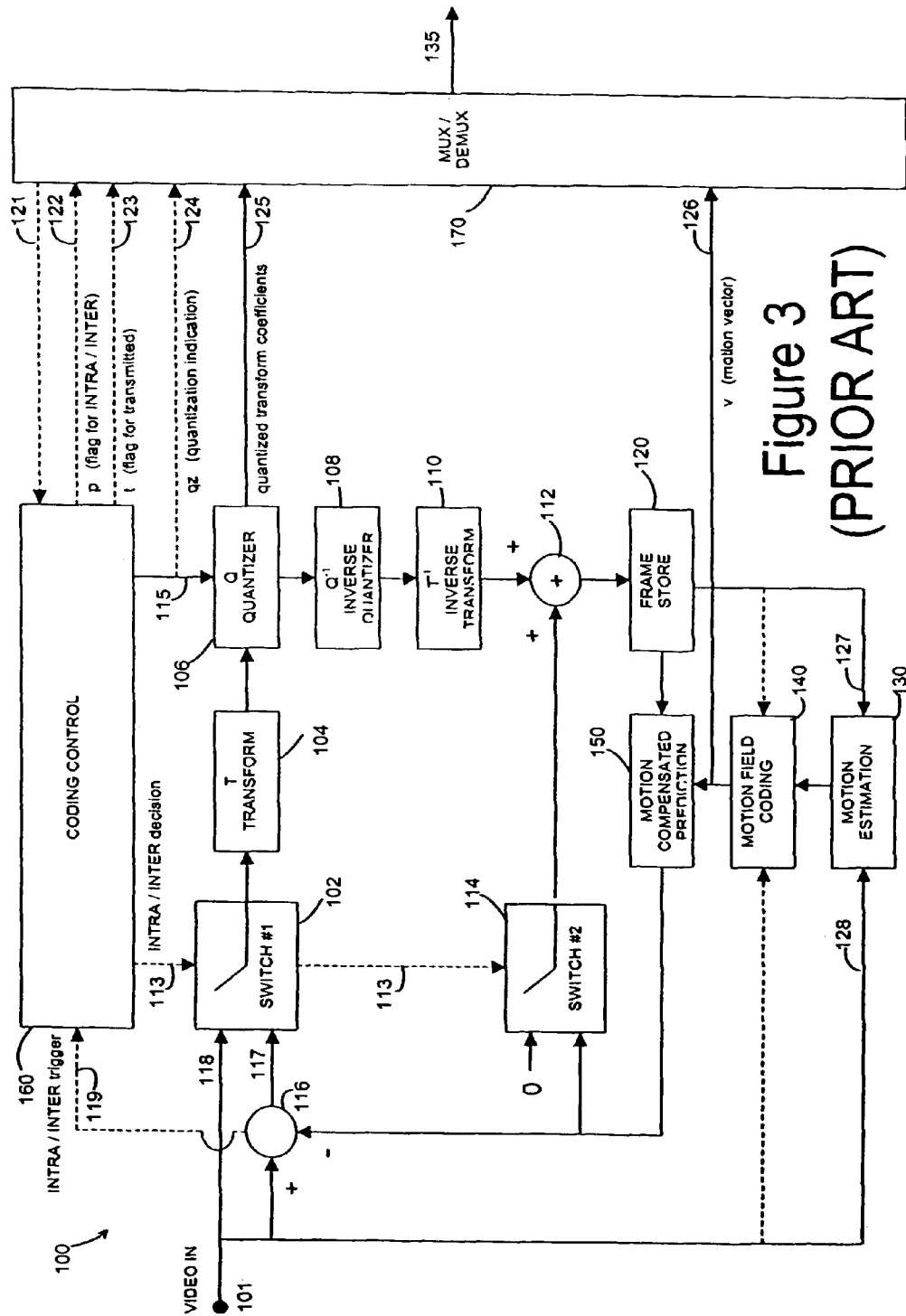


Figure 3
(PRIOR ART)

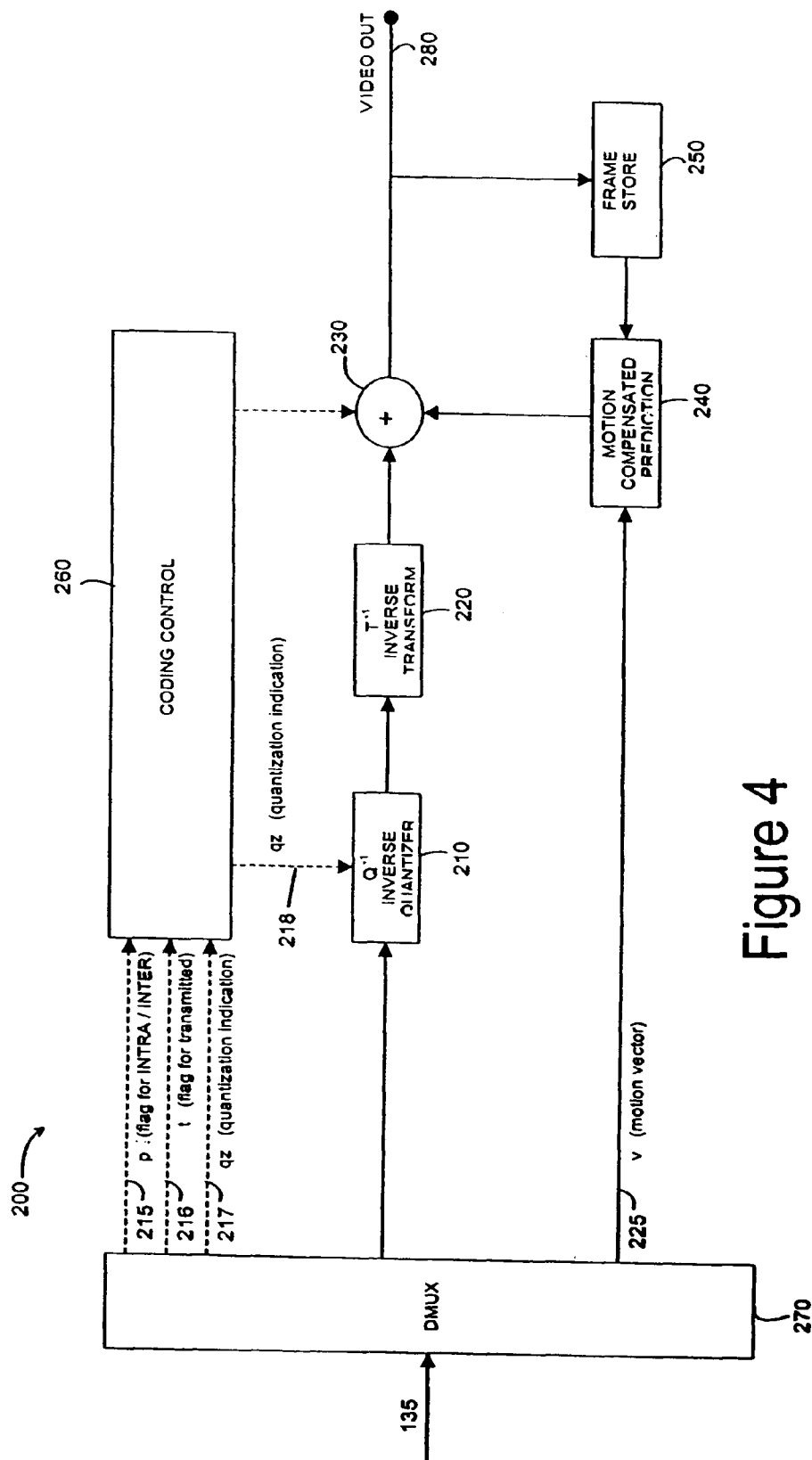


Figure 4
(PRIOR ART)

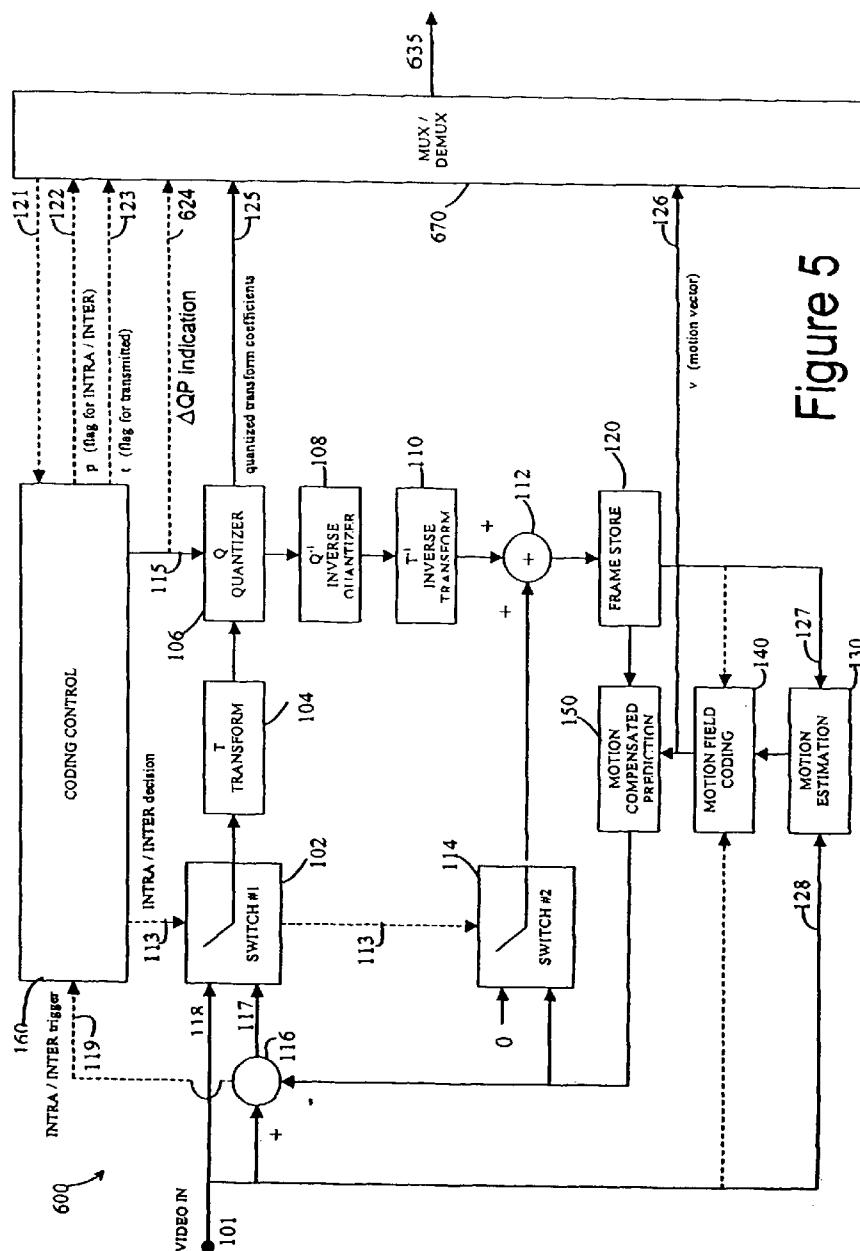


Figure 5

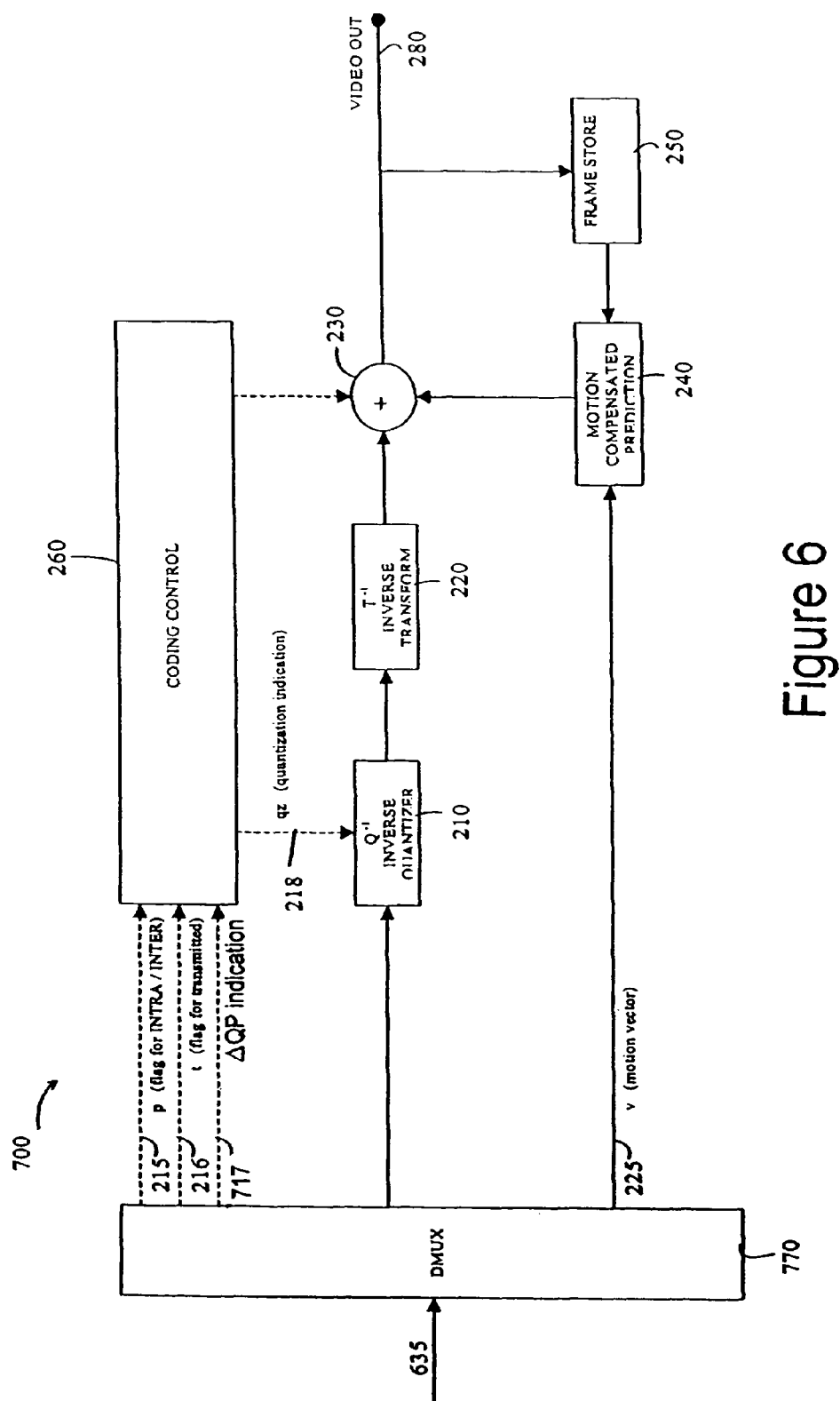


Figure 6

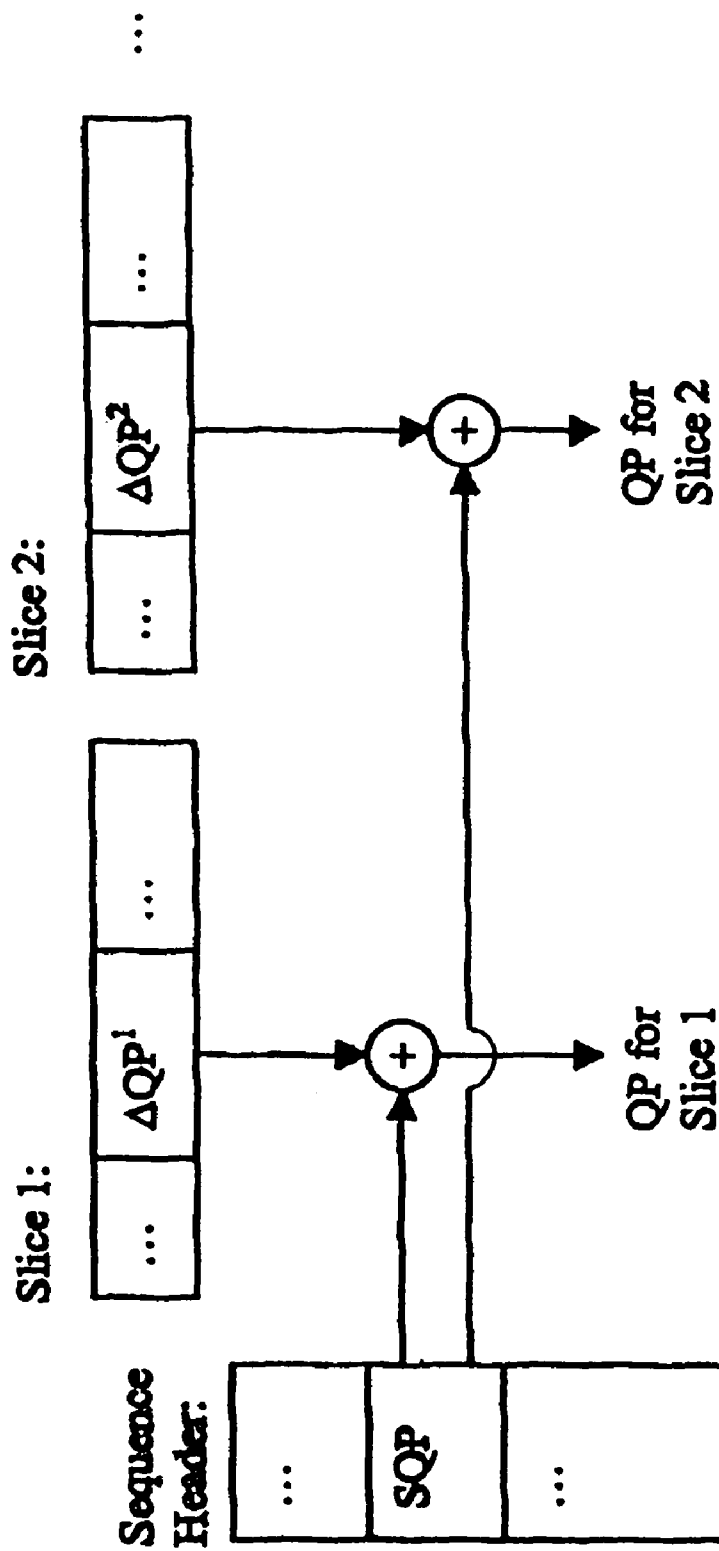


Figure 7

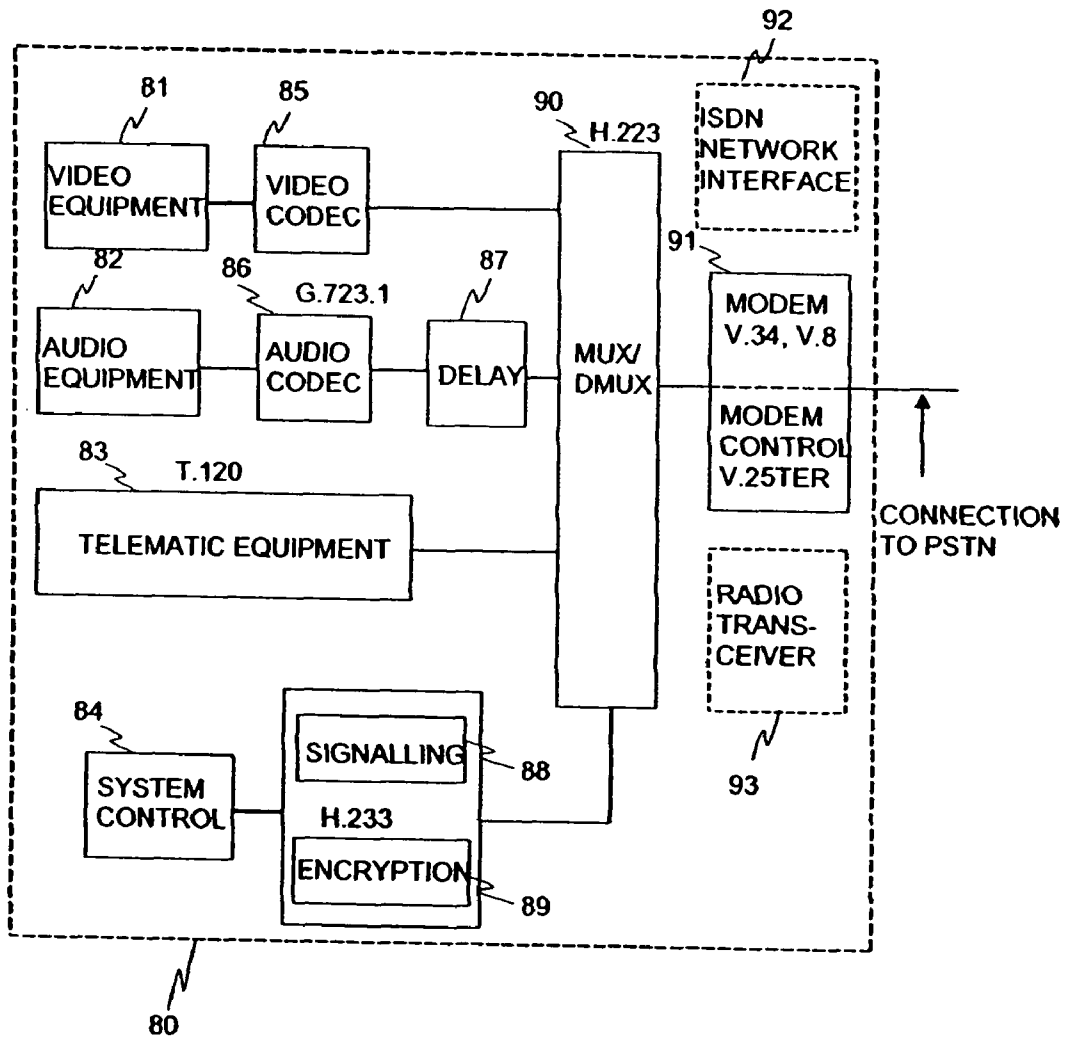


Figure 8

METHOD AND DEVICE FOR INDICATING QUANTIZER PARAMETERS IN A VIDEO CODING SYSTEM

This application is a divisional application of and claims priority to a U.S. patent application Ser. No. 10/421,629, filed Apr. 23, 2003, now U.S. Pat. No. 7,263,125 which is based on and claims priority to U.S. provisional application No. 60/374,667, filed Apr. 23, 2002.

FIELD OF THE INVENTION

The invention relates to a method, an encoder, a decoder and a device for the coding of digital video. More specifically, the invention relates to the indication of quantization parameter (QP) values in a video coding system.

BACKGROUND OF THE INVENTION

Digital video sequences, like ordinary motion pictures recorded on film, comprise a sequence of still images, the illusion of motion being created by displaying the images one after the other, typically at a rate between 15 and 30 frames per second.

Each frame of an uncompressed digital video sequence comprises an array of image pixels. In a commonly used digital video format, known as the Quarter Common Interchange Format (QCIF), a frame comprises an array of 176×144 pixels (i.e. 25,344 pixels). In turn, each pixel is represented by a certain number of bits, which carry information about the luminance and/or colour content of the region of the image corresponding to the pixel. Commonly, a so-called YUV colour model is used to represent the luminance and chrominance content of the image. The luminance, or Y, component represents the intensity (brightness) of the image, while the colour content of the image is represented by two chrominance or colour difference components, labelled U and V.

Colour models based on a luminance/chrominance representation of image content provide certain advantages compared with colour models that are based on a representation involving primary colours (that is Red, Green and Blue, RGB). The human visual system is more sensitive to intensity variations than it is to colour variations and YUV colour models exploit this property by using a lower spatial resolution for the chrominance components (U, V) than for the luminance component (Y). In this way, the amount of information needed to code the colour information in an image can be reduced with an acceptable reduction in image quality.

The lower spatial resolution of the chrominance components is usually attained by sub-sampling. Typically, each frame of a video sequence is divided into so-called "macroblocks", which comprise luminance (Y) information and associated chrominance (U, V) information, which is spatially sub-sampled. FIG. 1 illustrates one way in which macroblocks can be formed. As shown in FIG. 1, a frame of a video sequence represented using a YUV color model, each component having the same spatial resolution. Macroblocks are formed by representing a region of 16×16 image pixels in the original image as four blocks of luminance information, each luminance block comprising an 8×8 array of luminance (Y) values and two, spatially corresponding, chrominance components (U and V) which are sub-sampled by a factor of two in both the horizontal and vertical directions to yield corresponding arrays of 8×8 chrominance (U, V) values. According to certain video coding recommendations, such as International Telecommunications Union (ITU-T)

recommendation H.26L, the block size used within the macroblocks can be other than 8×8 , for example 4×8 or 4×4 (see T. Wiegand, "Joint Model Number 1", Doc. JVT-A003, Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, January 2002, Sections 2.2 and 2.3). ITU-T recommendation H.26L also allows macroblocks to be organized together to form so-called "slices". More specifically, each slice is formed from a number of consecutive macroblocks in coding order and is encoded in such a way that it can be decoded independently without making reference to any other slice of the same frame. This arrangement is advantageous, as it tends to limit the propagation of artefacts in the decoded video that may arise due to transmission errors. While there is no specific limitation on the way in which slices may be constructed, one straightforward scheme is to group all the macroblocks in a single row of a frame together as a slice. This arrangement, together with the division of a QCIF format image into 16×16 macroblocks is illustrated in FIG. 2.

As can be seen from FIG. 2, a QCIF image comprises 11×9 macroblocks (in this case grouped into 9 slices of 11 consecutive macroblocks each). If the luminance blocks and chrominance blocks are represented with 8 bit resolution (that is by numbers in the range 0 to 255), the total number of bits required per macroblock is $(16 \times 16 \times 8) + 2 \times (8 \times 8 \times 8) = 3072$ bits. The number of bits needed to represent a video frame in QCIF format is thus $99 \times 3072 = 304,128$ bits. This means that the amount of data required to transmit/record/display an uncompressed video sequence in QCIF format, represented using a YUV colour model, at a rate of 30 frames per second, is more than 9 Mbps (million bits per second). This is an extremely high data rate and is impractical for use in video recording, transmission and display applications because of the very large storage capacity, transmission channel capacity and hardware performance required.

If video data is to be transmitted in real-time over a fixed line network such as an ISDN (Integrated Services Digital Network) or a conventional PSTN (Public Switched Telephone Network), the available data transmission bandwidth is typically of the order of 64 kbits/s. In mobile videotelephony, where transmission takes place at least in part over a radio communications link, the available bandwidth can be as low as 20 kbits/s. This means that a significant reduction in the amount of information used to represent video data must be achieved in order to enable transmission of digital video sequences over low bandwidth communication networks. For this reason, video compression techniques have been developed which reduce the amount of information transmitted while retaining an acceptable image quality.

Video compression methods are based on reducing the redundant and perceptually irrelevant parts of video sequences. The redundancy in video sequences can be categorised into spatial, temporal and spectral redundancy. "Spatial redundancy" is the term used to describe the correlation (similarity) between neighbouring pixels within a frame. The term "temporal redundancy" expresses the fact that objects appearing in one frame of a sequence are likely to appear in subsequent frames, while "spectral redundancy" refers to the correlation between different colour components of the same image.

There is often a significant amount of spatial redundancy between the pixels that make up each frame of a digital video sequence. In other words, the value of any pixel within a frame of the sequence is substantially the same as the value of other pixels in its immediate vicinity. Typically, video coding systems reduce spatial redundancy using a technique known as "block-based transform coding", in which a mathematical transformation, such as a two-dimensional Discrete Cosine

Transform (DCT), is applied to blocks of image pixels. This transforms the image data from a representation comprising pixel values to a form comprising a set of coefficient values representative of spatial frequency components. This alternative representation of the image data reduces spatial redundancy significantly and thereby produces a more compact representation of the image data.

Frames of a video sequence which are compressed using block-based transform coding, without reference to any other frame within the sequence, are referred to as INTRA-coded or I-frames.

Generally, video coding systems not only reduce the spatial redundancy within individual frames of a video sequence, but also make use of a technique known as "motion-compensated prediction", to reduce the temporal redundancy in the sequence. Using motion-compensated prediction, the image content of some (often many) of the frames in a digital video sequence is "predicted" from one or more other frames in the sequence, known as "reference" frames. Prediction of image content is achieved by tracking the motion of objects or regions of an image between a frame to be coded (compressed) and the reference frame(s) using "motion vectors". As in the case of INTRA-coding, motion compensated prediction of a video frame is typically performed macroblock-by-macroblock.

Frames of a video sequence, compressed using motion-compensated prediction, are generally referred to as INTER-coded or P-frames. Motion-compensated prediction alone rarely provides a sufficiently precise representation of the image content of a video frame and therefore it is typically necessary to provide a so-called "prediction error" (PE) frame with each INTER-coded frame. The prediction error frame represents the difference between a decoded version of the INTER-coded frame and the image content of the frame to be coded. More specifically, the prediction error frame comprises values that represent the difference between pixel values in the frame to be coded and corresponding reconstructed pixel values formed on the basis of a predicted version of the frame in question. Consequently, the prediction error frame has characteristics similar to a still image and block-based transform coding can be applied in order to reduce its spatial redundancy and hence the amount of data (number of bits) required to represent it.

In order to illustrate the operation of a video coding system in greater detail, reference will now be made to FIGS. 3 and 4. FIG. 3 is a schematic diagram of a generic video encoder that employs a combination of INTRA- and INTER-coding to produce a compressed (encoded) video bit-stream. A corresponding decoder is illustrated in FIG. 4 and will be described later in the text.

The video encoder **100** comprises an input **101** for receiving a digital video signal from a camera or other video source (not shown). It also comprises a transformation unit **104** which is arranged to perform a block-based discrete cosine transform (DCT), a quantizer **106**, an inverse quantizer **108**, an inverse transformation unit **110**, arranged to perform an inverse block-based discrete cosine transform (IDCT), combiners **112** and **116**, and a frame store **120**. The encoder further comprises a motion estimator **130**, a motion field coder **140** and a motion compensated predictor **150**. Switches **102** and **114** are operated co-operatively by control manager **160** to switch the encoder between an INTRA-mode of video encoding and an INTER-mode of video encoding. The encoder **100** also comprises a video multiplex coder **170** which forms a single bit-stream from the various types of information produced by the encoder **100** for further trans-

mission to a remote receiving terminal or, for example, for storage on a mass storage medium, such as a computer hard drive (not shown).

Encoder **100** operates as follows. Each frame of uncompressed video provided from the video source to input **101** is received and processed macroblock by macroblock, preferably in raster-scan order. When the encoding of a new video sequence starts, the first frame to be encoded is encoded as an INTRA-coded frame. Subsequently, the encoder is programmed to code each frame in INTER-coded format, unless one of the following conditions is met: 1) it is judged that the current macroblock of the frame being coded is so dissimilar from the pixel values in the reference frame used in its prediction that excessive prediction error information is produced, in which case the current macroblock is coded in INTRA-coded format; 2) a predefined INTRA frame repetition interval has expired; or 3) feedback is received from a receiving terminal indicating a request for a frame to be provided in INTRA-coded format.

Operation of the encoder **100** in INTRA-coding mode will now be described. In INTRA-coding mode, the control manager **160** operates the switch **102** to accept video input from input line **118**. The video signal input is received macroblock by macroblock and the blocks of luminance and chrominance values which make up each macroblock are passed to the DCT transformation block **104**. Here a 2-dimensional discrete cosine transform is performed and a 2-dimensional array of DCT coefficients is formed for each block.

The DCT coefficients for each block are passed to the quantizer **106**, where they are quantized using a quantization parameter QP. Selection of the quantization parameter QP is controlled by the control manager **160** via control line **115**.

In more detail, quantization of the DCT coefficients is performed by dividing each coefficient value by the quantization parameter QP and rounding the result to the nearest integer. In this way the quantization process yields a set of quantized DCT coefficient values that have a reduced numerical precision compared with the coefficient values originally generated by DCT transformation block **104**. Thus, in general, each of the quantized DCT coefficients may be represented by a smaller number of data bits than required to represent the corresponding coefficient before quantization. Furthermore, certain DCT coefficients are reduced to zero by the quantization process, thus reducing the number of coefficients that must be coded. Both these effects result in a reduction in the amount of data (i.e. data bits) required to represent the DCT coefficients for an image block. Thus, quantization provides a further mechanism by which the amount of data required to represent each image of the video sequence can be reduced. It also introduces an irreversible loss of information, which leads to a corresponding reduction in image quality. While this reduction in image quality may not always be desirable, quantization of DCT coefficient values does provide the possibility to adjust the number of bits required to encode a video sequence to take into account e.g. the bandwidth available for transmission of the encoded sequence or the desired quality of the coded video. More specifically, by increasing the value of QP used to quantize the DCT coefficients, a lower quality, but more compact representation of the video sequence can be created. Conversely, by reducing the value of QP, a higher quality but less compressed encoded bit-stream can be formed.

The quantized DCT coefficients for each block are passed from the quantizer **106** to the video multiplex coder **170**, as indicated by line **125** in FIG. 1. The video multiplex coder **170** orders the quantized transform coefficients for each block using a zigzag scanning procedure, thereby converting the

two-dimensional array of quantized transform coefficient values into a one-dimensional array. Typically, the video multiplex coder **170** next represents each non-zero quantized coefficient in the one-dimensional array by a pair of values, referred to as level and run, level being the value of the quantized coefficient and run being the number of consecutive zero-valued coefficients preceding the coefficient in question. The run and level values are further compressed using entropy coding. For example, a method such as variable length coding (VLC) may be used to produce a set of variable length codewords representative of each (run, level) pair.

Once the (run, level) pairs have been entropy (e.g. variable length) coded, the video multiplex coder **170** further combines them with control information, also entropy coded, for example, using a variable length coding method appropriate for the kind of information in question, to form a single compressed bit-stream of coded image information **135**. It is this bit-stream, including the variable length codewords representative of the (run, level) pairs and control information relating, among other things, to the quantization parameter QP used for quantizing the DCT coefficients, which is transmitted from the encoder.

A locally decoded version of the macroblock is also formed in the encoder **100**. This is done by passing the quantized transform coefficients for each block, output by quantizer **106**, through inverse quantizer **108** and applying an inverse DCT transform in inverse transformation block **110**. Inverse quantization is performed by reversing the quantization operation performed in quantizer **106**. More specifically, inverse quantizer **108** attempts to recover the original DCT coefficient values for a given image block by multiplying each quantized DCT coefficient value by quantization parameter QP. Because of the rounding operation performed as part of the quantization process in quantizer **106**, it is generally not possible to recover the original DCT coefficient values exactly. This results in a discrepancy between the recovered DCT coefficient values and those originally produced by DCT transformation block **104** (hence the irreversible loss of information referred to above).

The operations performed by inverse quantizer **108** and inverse transformation block **110** yield a reconstructed array of pixel values for each block of the macroblock. The resulting decoded image data is input to combiner **112**. In INTRA-coding mode, switch **114** is set so that the input to the combiner **112** via switch **114** is zero. In this way, the operation performed by combiner **112** is equivalent to passing the decoded image data unaltered.

As subsequent macroblocks of the current frame are received and undergo the previously described encoding and local decoding steps in blocks **104**, **106**, **108**, **110** and **112**, a decoded version of the INTRA-coded frame is built up in frame store **120**. When the last macroblock of the current frame has been INTRA-coded and subsequently decoded, the frame store **120** contains a completely decoded frame, available for use as a prediction reference frame in coding a subsequently received video frame in INTER-coded format. The flag indicating INTRA or INTER-code format is provided in line **122**.

Operation of the encoder **100** in INTER-coding mode will now be described. In INTER-coding mode, the control manager **160** operates switch **102** to receive its input from line **117**, which comprises the output of combiner **116**. The combiner **116** receives the video input signal macroblock by macroblock from input **101**. As combiner **116** receives the blocks of luminance and chrominance values which make up the macroblock, it forms corresponding blocks of prediction error information. The prediction error information repre-

sents the difference between the block in question and its prediction, produced in motion compensated prediction block **150**. More specifically, the prediction error information for each block of the macroblock comprises a two-dimensional array of values, each of which represents the difference between a pixel value in the block of luminance or chrominance information being coded and a decoded pixel value obtained by forming a motion-compensated prediction for the block, according to the procedure described below.

The prediction error information for each block of the macroblock is passed to DCT transformation block **104**, which performs a two-dimensional discrete cosine transform on each block of prediction error values to produce a two-dimensional array of DCT transform coefficients for each block.

The transform coefficients for each prediction error block are passed to quantizer **106** where they are quantized using a quantization parameter QP, in a manner analogous to that described above in connection with operation of the encoder in INTRA-coding mode. Again, selection of the quantization parameter QP is controlled by the control manager **160** via control line **115**. The accuracy of prediction error coding can be adjusted depending on the available bandwidth and/or required quality of the coded video. In a typical Discrete Cosine Transform (DCT) based system this is done by varying the Quantizer Parameter (QP) used in quantizing the DCT coefficients the DCT coefficients to a specific accuracy.

The quantized DCT coefficients representing the prediction error information for each block of the macroblock are passed from quantizer **106** to video multiplex coder **170**, as indicated by line **125** in FIG. 1. As in INTRA-coding mode, the video multiplex coder **170** orders the transform coefficients for each prediction error block using a zigzag scanning procedure and then represents each non-zero quantized coefficient as a (run, level) pair. It further compresses the (run, level) pairs using entropy coding, in a manner analogous to that described above in connection with INTRA-coding mode. Video multiplex coder **170** also receives motion vector information (described in the following) from motion field coding block **140** via line **126** and control information (e.g. including an indication of the quantization parameter QP) from control manager **160**. It entropy codes the motion vector information and control information and forms a single bit-stream of coded image information, **135** comprising the entropy coded motion vector, prediction error and control information. The indication, qz, of the quantization parameter QP is provided to multiplex coder **170** via line **124**.

The quantized DCT coefficients representing the prediction error information for each block of the macroblock are also passed from quantizer **106** to inverse quantizer **108**. Here they are inverse quantized, in a manner analogous to that previously described in connection with operation of the encoder in INTRA-coding mode. In INTER-coding mode, the quality of the encoded video bit-stream and the number of bits required to represent the video sequence can be adjusted by varying the degree of quantization applied to the DCT coefficients representing the prediction error information.

The resulting blocks of inverse quantized DCT coefficients are applied to inverse DCT transform block **110**, where they undergo inverse DCT transformation to produce locally decoded blocks of prediction error values. The locally decoded blocks of prediction error values are then input to combiner **112**. In INTER-coding mode, switch **114** is set so that the combiner **112** also receives predicted pixel values for each block of the macroblock, generated by motion-compensated prediction block **150**. The combiner **112** combines each of the locally decoded blocks of prediction error values with

a corresponding block of predicted pixel values to produce reconstructed image blocks and stores them in frame store **120**.

As subsequent macroblocks of the video signal are received from the video source and undergo the previously described encoding and decoding steps in blocks **104**, **106**, **108**, **110**, **112**, a decoded version of the frame is built up in frame store **120**. When the last macroblock of the frame has been processed, the frame store **120** contains a completely decoded frame, available for use as a prediction reference frame in encoding a subsequently received video frame in INTER-coded format.

Formation of a prediction for a macroblock of the current frame will now be described. Any frame encoded in INTER-coded format requires a reference frame for motion-compensated prediction. This means, necessarily, that when encoding a video sequence, the first frame to be encoded, whether it is the first frame in the sequence, or some other frame, must be encoded in INTRA-coded format. This, in turn, means that when the video encoder **100** is switched into INTER-coding mode by control manager **160**, a complete reference frame, formed by locally decoding a previously encoded frame, is already available in the frame store **120** of the encoder. In general, the reference frame is formed by locally decoding either an INTRA-coded frame or an INTER-coded frame.

The first step in forming a prediction for a macroblock of the current frame is performed by motion estimation block **130**. The motion estimation block **130** receives the blocks of luminance and chrominance values which make up the current macroblock of the frame to be coded via line **128**. It then performs a block matching operation in order to identify a region in the reference frame which corresponds substantially with the current macroblock. In order to perform the block matching operation, motion estimation block accesses reference frame data stored in frame store **120** via line **127**. More specifically, motion estimation block **130** performs block-matching by calculating difference values (e.g. sums of absolute differences) representing the difference in pixel values between the macroblock under examination and candidate best-matching regions of pixels from a reference frame stored in the frame store **120**. A difference value is produced for candidate regions at all possible positions within a predefined search region of the reference frame and motion estimation block **130** determines the smallest calculated difference value. The offset between the macroblock in the current frame and the candidate block of pixel values in the reference frame that yields the smallest difference value defines the motion vector for the macroblock in question.

Once the motion estimation block **130** has produced a motion vector for the macroblock, it outputs the motion vector to the motion field coding block **140**. The motion field coding block **140** approximates the motion vector received from motion estimation block **130** using a motion model comprising a set of basis functions and motion coefficients. More specifically, the motion field coding block **140** represents the motion vector as a set of motion coefficient values which, when multiplied by the basis functions, form an approximation of the motion vector. Typically, a translational motion model having only two motion coefficients and basis functions is used, but motion models of greater complexity may also be used.

The motion coefficients are passed from motion field coding block **140** to motion compensated prediction block **150**. Motion compensated prediction block **150** also receives the best-matching candidate region of pixel values identified by motion estimation block **130** from frame store **120**. Using the approximate representation of the motion vector generated by

motion field coding block **140** and the pixel values of the best-matching candidate region from the reference frame, motion compensated prediction block **150** generates an array of predicted pixel values for each block of the macroblock. Each block of predicted pixel values is passed to combiner **116** where the predicted pixel values are subtracted from the actual (input) pixel values in the corresponding block of the current macroblock, thereby forming a set of prediction error blocks for the macroblock.

Operation of the video decoder **200**, shown in FIG. 2 will now be described. The decoder **200** comprises a video multiplex decoder **270**, which receives an encoded video bit-stream **135** from the encoder **100** and demultiplexes it into its constituent parts, an inverse quantizer **210**, an inverse DCT transformer **220**, a motion compensated prediction block **240**, a frame store **250**, a combiner **230**, a control manager **260**, and an output **280**.

The control manager **260** controls the operation of the decoder **200** in response to whether an INTRA- or an INTER-coded frame is being decoded. An INTRA/INTER trigger control signal, which causes the decoder to switch between decoding modes is derived, for example, from picture type information associated with each compressed video frame received from the encoder. The INTRA/INTER trigger control signal is extracted from the encoded video bit-stream by the video multiplex decoder **270** and is passed to control manager **260** via control line **215**.

Decoding of an INTRA-coded frame is performed macroblock-by-macroblock. The video multiplex decoder **270** separates the encoded information for the blocks of the macroblock from possible control information relating to the macroblock in question. The encoded information for each block of an INTRA-coded macroblock comprises variable length codewords representing the VLC coded level and run values for the non-zero quantized DCT coefficients of the block. The video multiplex decoder **270** decodes the variable length codewords using a variable length decoding method corresponding to the encoding method used in the encoder **100** and thereby recovers the (run, level) pairs. It then reconstructs the array of quantized transform coefficient values for each block of the macroblock and passes them to inverse quantizer **210**. Any control information relating to the macroblock is also decoded in the video multiplex decoder using an appropriate decoding method and is passed to control manager **260**. In particular, information relating to the level of quantization applied to the transform coefficients (i.e. quantization parameter QP) is extracted from the encoded bit-stream by video multiplex decoder **270** and provided to control manager **260** via control line **217**. The control manager, in turn, conveys this information to inverse quantizer **210** via control line **218**. Inverse quantizer **210** inverse quantizes the quantized DCT coefficients for each block of the macroblock according to the control information relating to quantization parameter QP and provides the now inverse quantized DCT coefficients to inverse DCT transformer **220**. The inverse quantization operation performed by inverse quantizer **210** is identical to that performed by inverse quantizer **108** in the encoder.

Inverse DCT transformer **220** performs an inverse DCT transform on the inverse quantized DCT coefficients for each block of the macroblock to form a decoded block of image information comprising reconstructed pixel values. The reconstructed pixel values for each block of the macroblock are passed via combiner **230** to the video output **280** of the decoder where, for example, they can be provided to a display device (not shown). The reconstructed pixel values for each block of the macroblock are also stored in frame store **250**.

Because motion-compensated prediction is not used in the encoding/decoding of INTRA coded macroblocks control manager **260** controls combiner **230** to pass each block of pixel values as such to the video output **280** and frame store **250**. As subsequent macroblocks of the INTRA-coded frame are decoded and stored, a decoded frame is progressively assembled in the frame store **250** and thus becomes available for use as a reference frame for motion compensated prediction in connection with the decoding of subsequently received INTER-coded frames.

INTER-coded frames are also decoded macroblock by macroblock. The video multiplex decoder **270** receives the encoded video bit-stream **135** and separates the encoded prediction error information for each block of an INTER-coded macroblock from encoded motion vector information and possible control information relating to the macroblock in question. As explained in the foregoing, the encoded prediction error information for each block of the macroblock typically comprises variable length codewords representing the level and run values for the non-zero quantized transform coefficients of the prediction error block in question. The video multiplex decoder **270** decodes the variable length codewords using a variable length decoding method corresponding to the encoding method used in the encoder **100** and thereby recovers the (run, level) pairs. It then reconstructs an array of quantized transform coefficient values for each prediction error block and passes them to inverse quantizer **210**. Control information relating to the INTER-coded macroblock is also decoded in the video multiplex decoder **270** using an appropriate decoding method and is passed to control manager **260**. Information relating to the level of quantization (QP) applied to the transform coefficients of the prediction error blocks is extracted from the encoded bit-stream and provided to control manager **260** via control line **217**. The control manager, in turn, conveys this information to inverse quantizer **210** via control line **218**. Inverse quantizer **210** inverse quantizes the quantized DCT coefficients representing the prediction error information for each block of the macroblock according to the control information relating to quantization parameter QP and provides the now inverse quantized DCT coefficients to inverse DCT transformer **220**. Again, the inverse quantization operation performed by inverse quantizer **210** is identical to that performed by inverse quantizer **108** in the encoder. The INTRA/INTER flag is provided in line **215**.

The inverse quantized DCT coefficients representing the prediction error information for each block are then inverse transformed in the inverse DCT transformer **220** to yield an array of reconstructed prediction error values for each block of the macroblock.

The encoded motion vector information associated with the macroblock is extracted from the encoded video bit-stream **135** by video multiplex decoder **270** and is decoded. The decoded motion vector information thus obtained is passed via control line **225** to motion compensated prediction block **240**, which reconstructs a motion vector for the macroblock using the same motion model as that used to encode the INTER-coded macroblock in encoder **100**. The reconstructed motion vector approximates the motion vector originally determined by motion estimation block **130** of the encoder. The motion compensated prediction block **240** of the decoder uses the reconstructed motion vector to identify the location of a region of reconstructed pixels in a prediction reference frame stored in frame store **250**. The region of pixels indicated by the reconstructed motion vector is used to form a prediction for the macroblock in question. More specifically, the motion compensated prediction block **240** forms

an array of pixel values for each block of the macroblock by copying corresponding pixel values from the region of pixels identified in the reference frame. These blocks of pixel values, derived from the reference frame, are passed from motion compensated prediction block **240** to combiner **230** where they are combined with the decoded prediction error information. In practice, the pixel values of each predicted block are added to corresponding reconstructed prediction error values output by inverse DCT transformer **220**. In this way, an array of reconstructed pixel values for each block of the macroblock is obtained. The reconstructed pixel values are passed to the video output **280** of the decoder and are also stored in frame store **250**.

As subsequent macroblocks of the INTER-coded frame are decoded and stored, a decoded frame is progressively assembled in the frame store **250** and thus becomes available for use as a reference frame for motion-compensated prediction of other INTER-coded frames.

As described above, typical video encoding and decoding systems (commonly referred to as video codecs) are based on motion compensated prediction and prediction error coding. Motion compensated prediction is obtained by analyzing and coding motion between video frames and reconstructing image segments using the motion information. Prediction error coding is used to code the difference between motion compensated image segments and corresponding segments of the original image. The accuracy of prediction error coding can be adjusted depending on the available bandwidth and required quality of the coded video. In a typical Discrete Cosine Transform (DCT) based system this is done by varying the quantization parameter (QP) used in quantizing the DCT coefficients to a specific accuracy.

It should be noted that, in order to stay in synchronization with the encoder, the decoder has to know the exact value of the QP used in the coded video sequence. Typically, the QP value is transmitted once per slice leading to increase in the number of bits needed to encode the image. (As previously explained, a slice contains part of the image and is coded independently from other slices in order to avoid propagation of possible transmission errors inside the picture). For example, if the coding of a single QP value takes 6 bits and 20 images, each divided into 10 slices, are transmitted every second, 1.2 kbps is spent for the QP information alone.

Prior art solutions (for example, the H.26L video coding recommendation presented in the document by T. Wiegand, "Joint Model Number 1", Doc. JVT-A003, Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, January 2002) code the picture/slice QP parameters independently with a fixed or variable length code. This leads to increased transmission bit-rate as described above. More specifically, according to H.26L Joint Model Number 1, the quantization parameter value QP used in quantizing the DCT coefficient values is typically indicated in the encoded bit-stream at the beginning of each picture (see T. Wiegand, "Joint Model Number 1", Doc. JVT-A003, Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, January 2002, Section 3.3.1). If the macroblocks within a frame are arranged in slices then the QP value is also indicated at the beginning of each slice of the frame (e.g. in an optional slice header portion of the encoded bit-stream). In both cases, the QP value is indicated as such or is coded using an appropriate variable length coding scheme. As stated above, it should be realized that this scheme is very costly in terms of the number of bits required to represent the quantization parameter information, particularly in situations where frames are divided into many slices and/or the available bandwidth available for transmission of the encoded video sequence is low. This is a particu-

larly significant problem in mobile video applications in which the encoded video bit-stream is transmitted over a radio communications link. In this situation, the bandwidth available for transmission of the encoded video bit-stream may be as low as 20 kbits/s and the QP information included in the bit-stream may represent a significant proportion of the overall available bandwidth.

Furthermore, according to H.26L, the value of QP may optionally be varied at the macroblock level by inserting a quantizer change parameter (Dquant) in the portion of the encoded bit-stream representative of the macroblock in question (see T. Wiegand, "Joint Model Number 1", Doc. JVT-A003, Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, January 2002, Section 3.4.7). This leads to a further increase in the amount of information in the encoded bit-stream that is devoted to the indication of QP related information.

In view of the foregoing, it should be realised that there is a significant need for an improved mechanism for indicating information relating to quantization parameter values in video coding systems.

SUMMARY OF THE INVENTION

The present invention improves prior art solutions for indicating QP related information by introducing a sequence level QP. This allows the encoder application to decide a video sequence dependent reference QP to be used in coding of the picture/slice QPs. Now, according to the invention, instead of coding the absolute values of picture/slice QPs, it is enough to code the difference between the reference sequence QP and the actually used picture/slice QP. In this way there is no need to transmit a full QP for every picture/slice, but a statistically smaller difference value is transmitted and used to reconstruct the picture/slice QP, thus leading to reduction in transmission bit-rate.

The bit-rate savings are most evident in the case of constant QP. In this case, it is sufficient to send just one bit per slice to indicate that the sequence QP is to be used when decoding the slice. For example, in the previously described example, the bit-rate for QP is reduced from 1.2 kbps to 0.2 kbps (now only one bit needs to be sent instead of six for each slice).

According to a first aspect of the invention, there is provided a method of encoding a digital video sequence, the method being applied in a video coding application to produce an encoded video bit-stream representative of the digital video sequence. The digital video sequence comprises a number of frames, each frame of the sequence comprising an array of pixels and is divided into a plurality of blocks, each block comprising a certain number of pixels. The method comprises encoding a frame of the digital video sequence by applying motion compensated prediction to blocks of pixels, thereby producing corresponding blocks of prediction error values. A transform coding technique is applied to the blocks of prediction error values to produce sets of transform coefficient values representative of the blocks of prediction error values and a level of quantization is applied to the sets of transform coefficient values to yield sets of quantized transform coefficient values. According to the invention, the method further comprises defining a default level of quantization to be used throughout encoding of the digital video sequence to quantize the sets of transform coefficient values.

Advantageously, the method according to the first aspect of the invention is also used to indicate quantization parameter (QP) values used to quantize sets of transform coefficient values representative of blocks of pixel values produced for frames coded in INTRA-coding mode in a manner analogous

to that described above for indicating the quantization parameter values used to quantize sets of transform coefficient values representative of prediction error values produced for frames coded in INTER-coding mode.

Advantageously, the default level of quantization to be used throughout encoding of the digital video sequence is specific to the video sequence being encoded. Alternatively, the default level of quantization is specific to the video coding application.

Preferably, an indication of the default level of quantization to be used throughout encoding of the digital video sequence is provided. Even more preferably, an indication of the default level of quantization is provided in the encoded bit-stream representative of the digital video sequence, advantageously, the encoded bit-stream, including the indication of the default level of quantization to be used throughout encoding of the digital video sequence, is transmitted to a video decoding device.

Advantageously, the default level of quantization to be used throughout encoding of the digital video sequence to quantize the sets of transform coefficient values can be updated during encoding of the digital video sequence and a further indication indicative of the updated default level of quantization is provided.

The indication of the updated default level of quantization is preferably transmitted to a video decoding device, advantageously, in the encoded bit-stream representative of the digital video sequence.

Advantageously, the level of quantization applied to said sets of transform coefficient values can be adjusted such that the actual level of quantization applied to the sets of transform coefficient values is different from the default level of quantization to be used throughout encoding of the video sequence. Preferably, the actual level of quantization applied is represented as a difference with respect to the default level of quantization. Advantageously, an indication of the difference with respect to the default level of quantization is provided in the encoded bit-stream representative of the digital video sequence.

In an embodiment of the video encoding method according to the first aspect of the invention, the level of quantization applied to the sets of transform coefficient values can be adjusted from one frame of the digital video sequence to another such that an actual level of quantization applied to the sets of transform coefficients for a particular frame of the digital video sequence is different from the default level of quantization to be used throughout encoding of the digital video sequence. Advantageously, in this embodiment, the actual level of quantization to be used in the particular frame is represented as a difference with respect to the default level of quantization and an indication of the difference with respect to the default level of quantization is provided in the encoded bit-stream representative of the digital video sequence.

In an alternative embodiment of the video encoding method according to the first aspect of the invention, the plurality of blocks into which a frame of the digital video sequence is divided are grouped into one or more segments and the level of quantization applied to the sets of transform coefficient values can be adjusted from one segment of a frame to another such that an actual level of quantization applied to the sets of transform coefficients for a particular segment of a frame is different from the default level of quantization to be used throughout encoding of the digital video sequence. Advantageously, in this alternative embodiment, the actual level of quantization to be used in the particular segment is represented as a difference with respect to

13

the default level of quantization and an indication of the difference with respect to the default level of quantization is provided in the encoded bit-stream representative of the digital video sequence.

Advantageously, if the default level of quantization is to be used to quantize all sets of transform coefficient values throughout the digital video sequence, an indication of the default level of quantization is provided together with an indication that said default level is to be used to quantize all sets of transform coefficient values throughout the digital video sequence.

According to a second aspect of the invention, there is provided a method of decoding an encoded digital video sequence, the method being applied in a video decoding application to produce a decoded digital video sequence. The digital video sequence comprises a number of frames, each frame of the sequence comprising an array of pixels and being divided into a plurality of blocks, each block comprising a certain number of pixels, frames of the digital video sequence having been encoded by applying motion compensated prediction to blocks of pixels to produce corresponding blocks of prediction error values and applying a transform coding technique to the blocks of prediction error values to produce sets of transform coefficient values representative of the blocks of prediction error values and applying a level of quantization to the sets of transform coefficient values to yield sets of quantized transform coefficient values representative of the blocks of prediction error values. According to the invention, the decoding method comprises defining a default level of inverse quantization to be used throughout decoding of the encoded digital video sequence to inverse quantize the sets of quantized transform coefficient values.

Advantageously, the default level of inverse quantization is identical to a default level of quantization used to quantize the sets of transform coefficient values during encoding of the video sequence.

Advantageously, the default level of inverse quantization defined to be used throughout decoding of the encoded digital video sequence is specific to the encoded video sequence being decoded. Alternatively, the default level of inverse quantization is specific to the video decoding application.

Advantageously, the decoding method comprises retrieving an indication of the default level of inverse quantization, preferably from a bit-stream representative of the encoded digital video sequence.

Advantageously, the default level of inverse quantization can be updated during decoding of the digital video sequence. Preferably, updating of the default level of inverse quantization is performed responsive to an indication of an updated default level of quantization used during encoding of the video sequence, retrieved from a bit-stream representative of the encoded digital video sequence. Alternatively, the default level of inverse quantization is updated responsive to an indication of an updated default level of quantization used during encoding of the video sequence, transmitted from a video encoding device.

Advantageously, the level of inverse quantization applied to the sets of quantized transform coefficient values can be adjusted such that an actual level of inverse quantization applied to the sets of quantized transform coefficient values is different from the default level of inverse quantization to be used throughout decoding of the encoded digital video sequence. In this case, the actual level of inverse quantization is determined by adding a difference value to the default level of inverse quantization, the difference value being representative of the difference between the default level of inverse quantization and the actual level of inverse quantization to be

14

applied. Preferably, an indication of the difference value is retrieved from a bit-stream representative of the encoded digital video sequence.

In an embodiment of the video decoding method according to the second aspect of the invention, the level of inverse quantization applied to the sets of quantized transform coefficient values is adjusted from one frame of the digital video sequence to another such that an actual level of inverse quantization applied to the sets of quantized transform coefficients for a particular frame of the digital video sequence is different from the default level of inverse quantization to be used throughout decoding of the encoded digital video sequence. Advantageously, in this embodiment, the actual level of inverse quantization to be used in the particular frame is determined by adding a frame-specific difference value to the default level of inverse quantization, the frame-specific difference value being representative of the difference between the default level of inverse quantization and the actual level of inverse quantization to be used in the particular frame. Preferably, an indication of said frame-specific difference value is retrieved from a bit-stream representative of the encoded digital video sequence.

In an alternative embodiment of the video decoding method according to the second aspect of the invention, the plurality of blocks into which a frame of said digital video sequence is divided are grouped into one or more segments and the level of inverse quantization applied to the sets of quantized transform coefficient values is adjusted from one segment of a frame to another such that an actual level of inverse quantization applied to the sets of quantized transform coefficients for a particular segment of a frame is different from the default level of inverse quantization to be used throughout decoding of the encoded digital video sequence. Advantageously, in this alternative embodiment, the actual level of inverse quantization to be used in the particular segment is determined by adding a segment-specific difference value to the default level of inverse quantization, the segment-specific difference value being representative of the difference between the default level of inverse quantization and the actual level of inverse quantization to be used in the particular segment. Preferably, an indication of the segment-specific difference value is retrieved from a bit-stream representative of the encoded digital video sequence.

According to a third aspect of the invention, there is provided an encoder for encoding a digital video sequence to produce an encoded video bit-stream representative of the digital video sequence, the digital video sequence comprising a number of frames, each frame of the sequence comprising an array of pixels and being divided into a plurality of blocks, each block comprising a certain number of pixels. The video encoder according to the third aspect of the invention is arranged to encode a frame of the digital video sequence by applying motion compensated prediction to blocks of pixels, thereby producing corresponding blocks of prediction error values. It is further arranged to apply a transform coding technique to blocks of prediction error values to produce sets of transform coefficient values representative of said blocks of prediction error values and to apply a level of quantization to said sets of transform coefficient values to yield sets of quantized transform coefficient values. According to the invention, the video encoder is further arranged to define a default level of quantization to be used throughout encoding of the digital video sequence to quantize the sets of transform coefficient values.

Advantageously, the encoder according to the third aspect of the invention is also arranged to indicate quantization parameter (QP) values used to quantize sets of transform

15

coefficient values representative of blocks of pixel values produced for frames coded in INTRA-coding mode in a manner analogous to that described above for indicating the quantization parameter values used to quantize sets of transform coefficient values representative of prediction error values produced for frames coded in INTER-coding mode.

Advantageously, the default level of quantization defined by the video encoder is specific to the video sequence being encoded.

Advantageously, the video encoder is further arranged to provide an indication of the default level of quantization in the encoded bit-stream representative of the digital video sequence. Even more advantageously, it is arranged to transmit the encoded bit-stream to a corresponding video decoder.

Advantageously the video encoder is further arranged to update the default level of quantization during encoding of the digital video sequence and to provide an indication of the updated default level of quantization. Preferably, the encoder is also arranged to transmit the indication of the updated default level of quantization to a corresponding video decoder. Advantageously, the encoder includes the indication of the updated default level of quantization in the encoded bit-stream representative of the digital video sequence.

Preferably, the video encoder is further arranged to adjust the level of quantization applied to the sets of transform coefficient values and thereby to apply an actual level of quantization that is different from the default level of quantization. Preferably, the video encoder is still further arranged to represent the actual level of quantization as a difference with respect to the default level of quantization and to provide an indication of the difference with respect to the default level of quantization in the encoded bit-stream representative of the digital video sequence.

In one embodiment, a video encoder according to the third aspect of the invention is arranged to adjust the level of quantization applied to the sets of transform coefficient values from one frame of the digital video sequence to another. In this way it is arranged to apply an actual level of quantization to the sets of transform coefficients for a particular frame that is different from the default level of quantization to be used throughout encoding of the digital video sequence. Advantageously, the video encoder according to this embodiment is further arranged to represent the actual level of quantization to be used in the particular frame as a difference with respect to the default level of quantization and to provide an indication of the difference with respect to the default level of quantization in the encoded bit-stream representative of the digital video sequence.

In an alternative embodiment, a video encoder according to the third aspect of the invention is further arranged to group the plurality of blocks into which a frame of the digital video sequence is divided into one or more segments and to adjust the level of quantization applied to the sets of transform coefficient values from one segment of a frame to another. In this way the video encoder is arranged to apply an actual level of quantization to the sets of transform coefficients for a particular segment of a frame that is different from the default level of quantization to be used throughout encoding of the digital video sequence. Advantageously, the video encoder according to this alternative embodiment is further arranged to represent the actual level of quantization to be used in the particular segment as a difference with respect to the default level of quantization and to provide an indication of the difference with respect to the default level of quantization in the encoded bit-stream representative of the digital video sequence.

16

In a particular embodiment, the video encoder is arranged to provide an indication of the default level of quantization and an indication that the default level is to be used to quantize all sets of transform coefficient values throughout the digital video sequence.

Advantageously, the video encoder according to the third aspect of the invention is provided in a multimedia terminal. Even more preferably, it is implemented in a radio telecommunications device.

According to a fourth aspect of the invention, there is provided a decoder for decoding an encoded digital video sequence to produce a decoded digital video sequence. The digital video sequence comprises a number of frames, each frame of the sequence comprising an array of pixels and being divided into a plurality of blocks, each block comprising a certain number of pixels, frames of the digital video sequence having been encoded by applying motion compensated prediction to blocks of pixels to produce corresponding blocks of prediction error values and applying a transform coding technique to the blocks of prediction error values to produce sets of transform coefficient values representative of said blocks of prediction error values and applying a level of quantization to the sets of transform coefficient values to yield sets of quantized transform coefficient values representative of the blocks of prediction error values. According to the invention, the video decoder is arranged to define a default level of inverse quantization to be used throughout decoding of the encoded digital video sequence to inverse quantize the sets of quantized transform coefficient values.

Preferably, the default level of inverse quantization is identical to a default level of quantization used to quantize the sets of transform coefficient values during encoding of the video sequence.

Advantageously, the default level of inverse quantization defined to be used throughout decoding of the encoded digital video sequence is specific to the encoded video sequence being decoded.

Advantageously, the video decoder is arranged to retrieve an indication of the default level of inverse quantization, preferably from a bit-stream representative of the encoded digital video sequence.

Advantageously, the video decoder is arranged to update the default level of inverse quantization during decoding of the digital video sequence, preferably by retrieving an indication of an updated default level of quantization from a bit-stream representative of the encoded digital video sequence. Alternatively, the video decoder is arranged to receive an indication of an updated default level of quantization transmitted from a video encoding device.

Preferably, the video decoder is arranged to adjust the level of inverse quantization applied to the sets of quantized transform coefficient values and to apply an actual level of inverse quantization to the sets of quantized transform coefficient values that is different from the default level of inverse quantization. Advantageously, the decoder is arranged to determine the actual level of inverse quantization by adding a difference value to the default level of inverse quantization, the difference value being representative of the difference between the default level of inverse quantization and the actual level of inverse quantization to be applied. Preferably, the video decoder is arranged to retrieve an indication of the difference value from a bit-stream representative of the encoded digital video sequence.

In one embodiment, a video decoder according to the fourth aspect of the invention is arranged to adjust the level of inverse quantization applied to the sets of quantized transform coefficient values from one frame of the digital video

17

sequence to another and to apply an actual level of inverse quantization to the sets of quantized transform coefficients for a particular frame of the digital video sequence that is different from the default level of inverse quantization to be used throughout decoding of the encoded digital video sequence. Advantageously, the decoder is arranged to determine the actual level of inverse quantization to be used in the particular frame by adding a frame-specific difference value to the default level of inverse quantization, the frame-specific difference value being representative of the difference between the default level of inverse quantization and the actual level of inverse quantization to be used in the particular frame. Preferably, the video decoder is arranged to retrieve an indication of the frame-specific difference value from a bit-stream representative of the encoded digital video sequence.

In an alternative embodiment, a video decoder according to the fourth aspect of the invention is arranged to decode an encoded video sequence in which the plurality of blocks into which a frame of said digital video sequence is divided are grouped into one or more segments and is further arranged to adjust the level of inverse quantization applied to the sets of quantized transform coefficient values from one segment of a frame to another and to apply an actual level of inverse quantization to the sets of quantized transform coefficients for a particular segment of a frame that is different from the default level of inverse quantization to be used throughout decoding of the encoded digital video sequence. Preferably, the decoder is arranged to determine the actual level of inverse quantization to be used in the particular segment by adding a segment-specific difference value to the default level of inverse quantization, the segment-specific difference value being representative of the difference between the default level of inverse quantization and the actual level of inverse quantization to be used in the particular segment. Preferably, the video decoder is arranged to retrieve an indication of the segment-specific difference value from a bit-stream representative of the encoded digital video sequence.

According to a fifth aspect of the invention, there is provided a multimedia terminal comprising an encoder according to the third aspect of the invention.

According to a sixth aspect of the invention, there is provided a multimedia terminal comprising a decoder according to the fourth aspect of the invention.

Preferably, the multimedia terminal according to the fifth and/or sixth aspects of the invention is a mobile multimedia terminal arranged to communicate with a mobile telecommunications network by means of a radio connection.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 illustrates the formation of a 16×16 macroblock according to prior art.

FIG. 2 illustrates subdivision of a QCIF picture into 16×16 macroblocks and grouping of consecutive macroblocks into slices.

FIG. 3 is a schematic block diagram of a generic video encoder according to prior art.

FIG. 4 is a schematic block diagram of a generic video decoder according to prior art and corresponding to the encoder shown in FIG. 3.

FIG. 5 is a schematic block diagram of a video encoder according to an embodiment of the invention.

FIG. 6 is a schematic block diagram of a video decoder according to an embodiment of the invention and corresponding to the encoder shown in FIG. 5.

FIG. 7 illustrates a decoding process according to one possible embodiment of the invention. Quantization param-

18

eters (QP) for each slice are obtained by adding the Sequence QP (SQP) to slice specific difference QP values (ΔQP^n).

FIG. 8 is a schematic block diagram of a multimedia communications terminal in which the method according to the invention may be implemented.

DETAILED DESCRIPTION OF THE INVENTION

In a preferred embodiment of the invention a video sequence specific quantization parameter (QP) is transmitted and used as a reference when coding and decoding the actual picture/slice quantization parameters. In this way there is no need to transmit a full QP for every picture/slice, but a statistically smaller difference value is transmitted and used to reconstruct the picture/slice QP, thus leading to reduction in transmission bit-rate.

Embodiments of the invention will now be described with reference to FIGS. 5 through 8.

FIG. 5 is a schematic block diagram of a video encoder 600 implemented according to the preferred embodiment of the invention. The structure of the video encoder shown in FIG. 5 is substantially identical to that of the prior art video encoder illustrated in FIG. 3, with appropriate modifications to those parts of the encoder that perform operations relating to the quantization of DCT transform coefficients and the signalling of quantization parameter (QP) values used in the video encoding process. All parts of the video encoder that implement functions and operate in a manner identical to the previously described prior art video encoder are identified with identical reference numbers. As the invention is particularly relevant to the signalling of quantization parameter (QP) values at the slice or frame level, it will be assumed throughout the following description that video encoder 600 according to the preferred embodiment of the invention is specifically adapted to apply a video coding method in which the frames of a video sequence to be coded are divided into macroblocks and the macroblocks are further grouped into slices, an indication of the quantization parameter being provided at the beginning of each frame and at the beginning of each new slice within a frame. An example of such a video coding method is the previously cited ITU-T H.26L video coding recommendation, as described in T. Wiegand, "Joint Model Number 1", Doc. JVT-A003, Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, January 2002. Alternatively, the method can be applied in a video coding system in which an indication of the quantization parameter is given only at the beginning of a frame. While the following detailed description is written specifically to illustrate application of the method according to the invention in the indication and signalling of slice level quantization parameters, it should be appreciated that the method can be applied in an exactly analogous manner to the representation of frame (picture) level quantization parameters.

Operation of video encoder 600 will now be considered in detail. When encoding a digital video sequence, encoder 600 operates in a manner similar to that previously described in connection with FIG. 3 to generate INTRA-coded and INTER-coded compressed video frames. As explained earlier in the text, in INTRA-coding mode, a Discrete Cosine Transform (DCT) is applied to each block of image data (pixel values) in order to produce a corresponding two-dimensional array of transform coefficient values. The DCT operation is performed in transform block 104 and the coefficients thus produced are subsequently passed to quantizer 106, where they are quantized. In INTER-coding mode, the DCT transform performed in block 104 is applied to blocks of prediction

error values. The transform coefficients produced as a result of this operation are also passed to quantizer 106 where they too are quantized.

According to the invention, when starting to encode a new video sequence, encoder 600 determines a default or reference level of quantization that is to be used throughout encoding of the video sequence to quantize the DCT coefficient values generated in quantizer 106. Throughout the following description, this default or reference level of quantization will be termed a "sequence level quantization parameter", or SQP for short. The choice of an SQP for a given video sequence is controlled by control manager 660 and, for example, may be based on consideration of the properties of the sequence to be encoded and the bandwidth available for transmission of the encoded bit-stream produced by the encoder.

In the preferred embodiment of the invention, encoder 600 determines SQP as the default or reference level of quantization to be used when operating in INTER-coding mode, i.e. in situations when the DCT coefficients generated in transform block 104 are representative of prediction error values. It should be appreciated that the method according to the invention can also be applied to the quantization of DCT coefficient values produced in traditional INTRA-coding mode utilizing no spatial prediction. However, given the different origins of the transform coefficients in INTRA- and INTER-coding modes (the DCT coefficients produced in INTRA-coding mode are derived from pixel values, while those generated in INTER-coding mode are produced by applying a DCT transform to prediction error values) it is unlikely that a single SQP value can be determined that is optimum for quantization of DCT coefficients in both INTRA- and INTER-coding modes. Therefore, in an embodiment in which the method according to the invention is used in both INTRA- and INTER-coding mode, two SQP values are preferably be used, one that provides the most efficient representation of QP value information in INTRA-coding mode and another that provides the most efficient representation of QP values used during INTER-coding. In all other respects, the method according to the invention may be applied in a directly analogous way in both INTRA- and INTER-coding modes. Of course, in an alternative embodiment, a single SQP value may be defined and used as the sequence level quantization parameter for both INTRA- and INTER-coded frames. This is a practical approach especially in modern video coding systems, such as the one described in T. Wiegand, "Joint Model Number 1", Doc. JVT-A003, Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, January 2002, where spatial prediction is applied to the INTRA coded macroblocks before coding the INTRA prediction error with DCT.

It should further be noted that as the majority of frames in a typical video sequence are encoded as INTER frames, the greatest saving in bit-rate is achieved by applying the method according to the invention to the representation of QP values in INTER-coding mode. Thus, in the preferred embodiment of the invention, a single SQP value is used, this value being indicative of a default or reference value of quantization to be used in quantizing DCT coefficients representative of prediction error values in INTER-coding mode.

Having determined the SQP value to be used for the sequence, control manager 660 provides an indication of the selected SQP value via control line 122 to video multiplex coder 670 which, in turn, inserts an indication of the SQP value into the bit-stream 635 of coded image information representative of the video sequence. Preferably, this indication is provided in a sequence header portion of the encoded video bit-stream 635.

Video encoder 600 then starts to encode the video sequence. As explained in connection with the description of prior art video encoder 100, illustrated in FIG. 3, the first frame of the sequence to be encoded is encoded in INTRA-format. As the SQP value defined according to the preferred embodiment of the invention is specific to the quantization of DCT coefficient values produced in INTER-coding mode, operation of encoder 600 in INTRA-coding mode is entirely analogous to that of prior art video encoder 100 and will not be considered in further detail here.

Once encoding of the first frame is complete, control manager 660 switches video encoder 600 into INTER-coding mode. In INTER-coding mode, switch 102 is operated to receive its input from line 117, which comprises the output of combiner 116. Combiner 116 receives the video input signal macroblock by macroblock from input 101 and forms a block of prediction error information for each block of the macroblock. The prediction error information for each block is passed to DCT transformer 104, which performs a two-dimensional discrete cosine transform on each block of prediction error values to produce a two-dimensional array of DCT transform coefficients for the block in question. The transform coefficients for each prediction error block are then passed to quantizer 106 where they are quantized, as previously described, using a quantization parameter QP. The rest of the INTER-coding process continues as previously described in connection with prior art video encoder 100.

As each macroblock is received, control manager 660 determines whether the macroblock currently being processed is the first macroblock of a slice. If it is, the control manager determines a quantization parameter value QP to be used in quantizing the DCT coefficient values generated in DCT transformer 104. It should be noted that it is possible to make an estimation about the QP based on the bit-budget allowed for the frame, bits already spent in the previous slices or the same frame and possibly the bits spent for slices in the previous frame. Having done this, control manager 660 determines a difference (ΔQP) between the previously defined sequence level quantization parameter value SQP and the actual QP value to be used for the slice in question. It then passes an indication of this difference via control line 624 to video multiplex coder 670, which further includes an indication of the difference value ΔQP in bit-stream 635. Preferably, this indication is provided in a slice header portion of the encoded video bit-stream 635 containing control information specific to the slice in question. This process is repeated until all slices of the current frame have been encoded in INTER-coded format, at which point the video encoder starts to encode the next frame of the video sequence.

A video decoder 700 implemented according to a preferred embodiment of the invention will now be described with reference to FIG. 6. The structure of the video decoder illustrated in FIG. 6 is substantially identical to that of the prior art video decoder shown in FIG. 4, with appropriate modifications to those parts of the decoder that perform operations relating to the inverse quantization of DCT transform coefficients. All parts of the video decoder that implement functions and operate in a manner identical to the previously described prior art video decoder are identified with identical reference numbers.

Here it is assumed that the video decoder of FIG. 6 corresponds to the encoder described in connection with FIG. 5 and is therefore capable of receiving and decoding the bit-stream 635 transmitted by encoder 600. As previously described, in the preferred embodiment of the invention, encoder 600 determines a sequence level quantization parameter SQP to be used in INTER-coding mode. Correspond-

ingly, decoder **700** is adapted to receive an indication of this SQP value and to use the sequence level quantization parameter SQP in the determination of inverse quantization parameters to be applied to blocks of quantized transform coefficient values (representative of prediction error values) received in the encoded bit-stream for INTER-coded frames. In an alternative embodiment of the invention, the same process may also be applied to quantized transform coefficient values extracted from the bit-stream for INTRA-coded frames. As explained above, in this alternative embodiment, an indication of two SQP values may be provided, one for INTRA-coded frames of the sequence and one for INTER-coded frames. In a further alternative embodiment, a single sequence level quantization parameter may be indicated for frames coded in both INTRA- and INTER-coding modes.

Operation of a video decoder according to the preferred embodiment of the invention will now be described in detail. Decoder **700** receives bit-stream **635** and separates it into its constituent parts. This operation is performed by video multiplex decoder **770**.

When starting to decode a new sequence, video multiplex decoder **770** first extracts information and parameters relating to the sequence as a whole from the sequence header portion of the received bit-stream **635**. As explained above in connection with the description of encoder **600**, according to the preferred embodiment of the invention, the sequence header portion of the bit-stream is modified to contain an indication of the sequence level quantization parameter SQP used in quantization of DCT coefficient values produced in INTER-coding mode. Video multiplex decoder extracts the indication of the SQP value from the bit-stream and, if it was encoded, for example using variable length coding, applies appropriate decoding to recover the SQP value. It then passes the SQP value to the decoder's control manager **760**, which stores it in the decoder's memory.

Video decoder **700** then starts to decode the encoded frames of the video sequence, the decoding of each frame commencing as soon as the video decoder starts receiving information relating to the frame in video bit-stream **635**. Video multiplex decoder **770** extracts an INTRA/INTER trigger control signal from picture type information associated with each compressed video frame received in the encoded bit-stream **635** and passes it to control manager **760** via control line **215**. The control manager **760** controls the operation of the decoder responsive to the INTRA/INTER trigger control signal, so as to switch the decoder into the correct decoding mode.

In the preferred embodiment of the invention, the decoding of frames coded in INTRA-format is performed in a manner analogous to that previously described in connection with the operation of prior art video decoder **200**. The decoding of frames encoded in INTER-format, on the other hand, proceeds as described in the following.

When it receives an indication that the next frame to be decoded is an INTER-coded frame, extracted from the received bit-stream by video multiplex decoder **770**, control manager **760** switches decoder **700** into INTER-mode. As explained in connection with the description of encoder **600**, according to the preferred embodiment of the invention, in which the macroblocks of each frame are grouped into slices, encoded bit-stream **635** comprises certain slice-specific control information, which includes an indication of a slice-specific QP value, represented as a difference value ΔQP with respect to the sequence level quantization parameter SQP. Advantageously, the control information specifically related to each slice is provided in the bit-stream as a header portion specific to the slice in question. On receiving such a portion of

the bit-stream, video multiplex decoder extracts the slice-specific control information from the slice header portion of the bit-stream and passes the indication of ΔQP for the slice, retrieved from the bit-stream, to control manager **760** via control line **717**.

Next, control manager **760** determines a level of inverse quantization to be applied to the quantized DCT coefficients of the macroblocks in the slice. This is done by combining the ΔQP value for the slice with the sequence specific quantization parameter SQP, previously received and stored in the decoder's memory. As described earlier in the text, the inverse quantization operation performed in the decoder involves multiplying each quantized DCT coefficient by a value equal to the level of quantization originally applied, that is, by the QP value used in the corresponding encoder to quantize the DCT coefficients. Thus, according to the preferred embodiment of the invention, control manager **760** determines the level of inverse quantization for the macroblocks of the slice by adding the received ΔQP value for the slice to SQP. It then passes this value to inverse quantizer **210** via control line **218**.

As encoded information for each macroblock of the slice is received in the bit-stream **635**, video multiplex decoder **770** separates the encoded prediction error information for each block of the macroblock from encoded motion vector information. It reconstructs the quantized DCT transform coefficients representative of the prediction error values for each block and passes them to inverse quantizer **210**. Inverse quantizer **210** then inverse quantizes the quantized DCT coefficients according to the slice QP constructed from the ΔQP and SQP values by control manager **760**. It then provides the inverse quantized DCT coefficients to inverse DCT transformer **220**. The rest of the decoding process continues as previously described in connection with prior art video decoder **200**.

The steps of receiving a slice-specific ΔQP value, combining ΔQP with SQP and inverse quantizing the quantized DCT coefficients for each block of the macroblocks within the slice is repeated for each slice of the frame until all slices of the current INTER-coded frame have been decoded. At this point video encoder **700** starts decoding the next frame of the encoded video sequence.

FIG. 7 illustrates the way in which slice-specific QP values are reconstructed according to the preferred embodiment of the invention. As can be seen from the figure, the process comprises the steps of:

1. retrieving a sequence level quantization parameter (SQP);
2. retrieving a picture or slice level difference quantization parameter (ΔQP);
3. adding the difference quantization parameters to the sequence level quantization parameter to obtain the quantization parameters for a picture or a slice;
4. constructing the received prediction error-coding coefficients by means of said picture or slice quantization parameter.

FIG. 8 presents a terminal device comprising video encoding and decoding equipment which may be adapted to operate in accordance with the present invention. More precisely, the figure illustrates a multimedia terminal **80** implemented according to ITU-T recommendation H.324. The terminal can be regarded as a multimedia transceiver device. It includes elements that capture, encode and multiplex multimedia data streams for transmission via a communications network, as well as elements that receive, de-multiplex, decode and display received multimedia content. ITU-T recommendation H.324 defines the overall operation of the terminal and refers to other recommendations that govern the

23

operation of its various constituent parts. This kind of multimedia terminal can be used in real-time applications such as conversational videotelephony, or non real-time applications such as the retrieval and/or streaming of video clips, for example from a multimedia content server in the Internet.

In the context of the present invention, it should be appreciated that the H.324 terminal shown in FIG. 8 is only one of a number of alternative multimedia terminal implementations suited to application of the inventive method. It should also be noted that a number of alternatives exist relating to the location and implementation of the terminal equipment. As illustrated in FIG. 8, the multimedia terminal may be located in communications equipment connected to a fixed line telephone network such as an analogue PSTN (Public Switched Telephone Network). In this case the multimedia terminal is equipped with a modem 91, compliant with ITU-T recommendations V.8, V.34 and optionally V.8 bis. Alternatively, the multimedia terminal may be connected to an external modem. The modem enables conversion of the multiplexed digital data and control signals produced by the multimedia terminal into an analogue form suitable for transmission over the PSTN. It further enables the multimedia terminal to receive data and control signals in analogue form from the PSTN and to convert them into a digital data stream that can be demultiplexed and processed in an appropriate manner by the terminal.

An H.324 multimedia terminal may also be implemented in such a way that it can be connected directly to a digital fixed line network, such as an ISDN (Integrated Services Digital Network). In this case the modem 91 is replaced with an ISDN user-network interface. In FIG. 8, this ISDN user-network interface is represented by alternative block 92.

H.324 multimedia terminals may also be adapted for use in mobile communication applications. If used with a wireless communication link, the modem 91 can be replaced with any appropriate wireless interface, as represented by alternative block 93 in FIG. 8. For example, an H.324/M multimedia terminal can include a radio transceiver enabling connection to the current 2nd generation GSM mobile telephone network, or the proposed 3rd generation UMTS (Universal Mobile Telephone System).

It should be noted that in multimedia terminals designed for two-way communication, that is for transmission and reception of video data, it is advantageous to provide both a video encoder and video decoder implemented according to the present invention. Such an encoder and decoder pair is often implemented as a single combined functional unit, referred to as a "codec".

A typical H.324 multimedia terminal will now be described in further detail with reference to FIG. 8.

The multimedia terminal 80 includes a variety of elements referred to as "terminal equipment". This includes video, audio and telematic devices, denoted generically by reference numbers 81, 82 and 83, respectively. The video equipment 81 may include, for example, a video camera for capturing video images, a monitor for displaying received video content and optional video processing equipment. The audio equipment 82 typically includes a microphone, for example for capturing spoken messages, and a loudspeaker for reproducing received audio content. The audio equipment may also include additional audio processing units. The telematic equipment 83, may include a data terminal, keyboard, electronic whiteboard or a still image transceiver, such as a fax unit.

The video equipment 81 is coupled to a video codec 85. The video codec 85 comprises a video encoder 600 and a corresponding video decoder 700 both implemented according to the invention (see FIGS. 5 and 6). The video codec 85

24

is responsible for encoding captured video data in an appropriate form for further transmission over a communications link and decoding compressed video content received from the communications network. In the example illustrated in FIG. 8, the video codec is implemented according to ITU-T recommendation H.26L, with appropriate modifications to implement the method according to the invention in both the encoder and the decoder of the video codec.

The terminal's audio equipment is coupled to an audio codec, denoted in FIG. 8 by reference number 86. Like the video codec, the audio codec comprises an encoder/decoder pair. It converts audio data captured by the terminal's audio equipment into a form suitable for transmission over the communications link and transforms encoded audio data received from the network back into a form suitable for reproduction, for example on the terminal's loudspeaker. The output of the audio codec is passed to a delay block 87. This compensates for the delays introduced by the video coding process and thus ensures synchronisation of audio and video content.

The system control block 84 of the multimedia terminal controls end-to-network signalling using an appropriate control protocol (signalling block 88) to establish a common mode of operation between a transmitting and a receiving terminal. The signalling block 88 exchanges information about the encoding and decoding capabilities of the transmitting and receiving terminals and can be used to enable the various coding modes of the video encoder. The system control block 84 also controls the use of data encryption. Information regarding the type of encryption to be used in data transmission is passed from encryption block 89 to the multiplexer/de-multiplexer (MUX/DMUX unit) 90.

During data transmission from the multimedia terminal, the MUX/DMUX unit 90 combines encoded and synchronised video and audio streams with data input from the telematic equipment 83 and possible control data, to form a single bit-stream. Information concerning the type of data encryption (if any) to be applied to the bit-stream, provided by encryption block 89, is used to select an encryption mode. Correspondingly, when a multiplexed and possibly encrypted multimedia bit-stream is being received, MUX/DMUX unit 90 is responsible for decrypting the bit-stream, dividing it into its constituent multimedia components and passing those components to the appropriate codec(s) and/or terminal equipment for decoding and reproduction.

If the multimedia terminal 80 is a mobile terminal, that is, if it is equipped with a radio transceiver 93, it will be understood by those skilled in the art that it may also comprise additional elements. In one embodiment it comprises a user interface having a display and a keyboard, which enables operation of the multimedia terminal 80 by a user, a central processing unit, such as a microprocessor, which controls the blocks responsible for different functions of the multimedia terminal, a random access memory RAM, a read only memory ROM, and a digital camera. The microprocessor's operating instructions, that is program code corresponding to the basic functions of the multimedia terminal 80, is stored in the read-only memory ROM and can be executed as required by the microprocessor, for example under control of the user. In accordance with the program code, the microprocessor uses the radio transceiver 93 to form a connection with a mobile communication network, enabling the multimedia terminal 80 to transmit information to and receive information from the mobile communication network over a radio path.

The microprocessor monitors the state of the user interface and controls the digital camera. In response to a user command, the microprocessor instructs the camera to record digi-

tal images into the RAM. Once an image is captured, or alternatively during the capturing process, the microprocessor segments the image into image segments (for example macroblocks) and uses the encoder to perform motion compensated encoding of the segments in order to generate a compressed image sequence, as explained in the foregoing description. A user may command the multimedia terminal 80 to display the captured images on its display or to send the compressed image sequence using the radio transceiver 93 to another multimedia terminal, a video telephone connected to a fixed line network (PSTN) or some other telecommunications device. In a preferred embodiment, transmission of image data is started as soon as the first segment is encoded so that the recipient can start a corresponding decoding process with a minimum delay.

Although described in the context of particular embodiments, it will be apparent to those skilled in the art that a number of modifications and various changes to these teachings may be made. Thus, while the invention has been particularly shown and described with respect to one or more preferred embodiments thereof, it will be understood by those skilled in the art that certain modifications or changes may be made therein without departing from the scope and spirit of the invention as set forth above.

In particular, according to a second possible embodiment of the invention, a sequence QP is not transmitted but an application specific constant is used as sequence QP instead.

In a third possible embodiment of the invention the sequence QP can be updated depending on the changing characteristics of the video sequence if a reliable way of transmitting the new sequence QP is available. The updated SQP value can be either included in the encoded bit-stream representative of the video sequence or can be transmitted directly from encoder to decoder in an associated control channel.

In a fourth possible embodiment of the invention, if QP is constant for the whole video sequence only the value of the sequence QP is transmitted together with the information that it should be used as the QP for all the pictures/slices.

What is claimed is:

1. A method of encoding a digital video sequence for use in a video coding application to produce an encoded video bit-stream representative of the digital video sequence, the digital video sequence comprising a number of frames, each frame of said sequence comprising an array of pixels divided into a plurality of blocks, each block comprising a certain number of said pixels, said method comprising:

encoding a frame of the digital video sequence by applying motion compensated prediction to blocks of pixels for producing corresponding blocks of prediction error values;

applying a transform coding technique to said blocks of prediction error values to produce sets of transform coefficient values representative of said blocks of prediction error values;

defining a default level of quantization for use in encoding of the digital video sequence to quantize the sets of transform coefficient values; and

providing an indication of the default level of quantization to a decoding process.

2. An encoding method according to claim 1, wherein said indication of the default level of quantization is provided in the encoded bit-stream representative of the digital video sequence.

3. An encoding method according to claim 2, wherein the encoded bit-stream is transmitted from a video encoding device to a corresponding video decoding device.

4. An encoding method according to claim 1, wherein the default level of quantization is updated during the encoding of the digital video sequence, said method further comprising providing an indication of the updated default level of quantization to a decoding process.

5. An encoding method according to claim 1, wherein said sets of transform coefficient values are quantized to yield sets of quantized transform coefficient values representative of said blocks of prediction error values based on a level of quantization different from the default level of quantization.

6. An encoding method according to claim 5, further comprising providing to a decoding process an indication of a difference between said level of quantization and the default level of quantization.

7. An encoding method according to claim 5, wherein said level of quantization is adjusted from one frame of the digital video sequence to another such that an actual level of quantization applied to the sets of transform coefficients for a particular frame of the digital video sequence is different from the default level of quantization.

8. An encoding method according to claim 7, wherein said actual level of quantization is representable as a difference with respect to the default level of quantization.

9. An encoding method according to claim 7, further comprising providing to a decoding process an indication of a difference between said actual level of quantization and the default level of quantization.

10. An encoding method according to claim 5, wherein the plurality of blocks of pixels, into which a frame of said digital video sequence is divided, are grouped into one or more segments and wherein the level of quantization applied to said sets of transform coefficient values is adjusted from one segment of a frame to another such that an actual level of quantization applied to the sets of transform coefficients for a particular segment of a frame is different from the default level of quantization.

11. An encoding method according to claim 10, further comprising providing to a decoding process an indication of a difference between said level of quantization and the default level of quantization.

12. A video encoder for encoding a digital video sequence to produce an encoded video bit-stream representative of the digital video sequence, the digital video sequence comprising a number of frames, each frame of said sequence comprising an array of pixels divided into a plurality of blocks, each block comprising a certain number of said pixels, said video encoder comprising:

means for encoding a frame of the digital video sequence by applying motion compensated prediction to blocks of pixels to provide corresponding blocks of prediction error values;

means for transforming said blocks of prediction error values for providing sets of transform coefficient values representative of said blocks of prediction error values; means for selecting a default level of quantization for quantizing said sets of transform coefficient values; and

means for providing an indication of said default level of quantization in the encoded bit-stream representative of the digital video sequence.

13. A video encoder according to claim 12, wherein said sets of transform coefficient values are quantized to yield sets of quantized transform coefficient values representative of said blocks of prediction error values based on a level of quantization different from the default level of quantization.

27

14. A video encoder according to claim 13, arranged to provide an indication of a difference between said level of quantization and the default level of quantization in the encoded bit-stream.

15. A video encoder according to claim 12, wherein the default level of quantization can be updated and the video encoder is arranged to provide an indication of the updated default level of quantization in the encoded bit-stream.

16. A video encoder according to claim 15, wherein said sets of transform coefficient values are quantized to yield sets of quantized transform coefficient values representative of said blocks of prediction error values based on a level of quantization different from the updated default level of quantization.

17. A video encoder according to claim 12, provided in a multimedia terminal.

18. A video encoder according to claim 12, provided in a radio telecommunications device.

19. A video encoder according to claim 13, wherein said level of quantization applied to said sets of transform coefficient values can be adjusted from one frame of the digital video sequence to another so as to apply an actual level of quantization to the sets of transform coefficients for a particular frame of the digital video sequence and wherein said actual level of quantization is different from the default level of quantization.

20. A video encoder according to claim 19, arranged to provide an indication of said difference with respect to the default level of quantization in the encoded bit-stream representative of the digital video sequence.

21. A video encoder according to claim 13, arranged to group the plurality of blocks into which a frame of said digital video sequence is divided into one or more segments and to

28

adjust the level of quantization applied to said sets of transform coefficient values from one segment of a frame to another so as to apply an actual level of quantization to the sets of transform coefficients for a particular segment of a frame, wherein said actual level of quantization is different from the default level of quantization, said video encoder further arranged to represent the actual level of quantization for use in the particular segment as a difference with respect to the default level of quantization.

22. A video encoder according to claim 21, further arranged to provide an indication of said difference with respect to the default level of quantization in the encoded bit-stream representative of the digital video sequence.

23. A video encoder for encoding a digital video sequence to produce an encoded video bit-stream representative of the digital video sequence, the digital video sequence comprising a number of frames, each frame of said sequence comprising an array of pixels divided into a plurality of blocks, each block comprising a certain number of said pixels, said video encoder comprising:

a module for encoding a frame of the digital video sequence by applying motion compensated prediction to blocks of pixels to provide corresponding blocks of prediction error values;

a module for transforming said blocks of prediction error values for providing sets of transform coefficient values representative of said blocks of prediction error values; and

a module for selecting a default level of quantization for quantizing said sets of transform coefficient values, wherein the default level is specific to the video coding applications.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 8,175,148 B2
APPLICATION NO. : 11/881367
DATED : May 8, 2012
INVENTOR(S) : Lainema

Page 1 of 4

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Delete the title page and substitute therefore with the attached title page consisting of corrected illustrative figure.

Drawings.

Sheets 5 and 6 should be deleted to be substituted with the attached sheets 5 and 6, as shown on the attached pages.

Signed and Sealed this
Thirtieth Day of September, 2014



Michelle K. Lee
Deputy Director of the United States Patent and Trademark Office

(12) **United States Patent**
Lainema

(10) **Patent No.:** US 8,175,148 B2
(45) **Date of Patent:** *May 8, 2012

- (54) METHOD AND DEVICE FOR INDICATING
QUANTIZER PARAMETERS IN A VIDEO
CODING SYSTEM

(75) Inventor: **Jani Lainema**, Irving, TX (US)

(73) Assignee: **Nokla Corporation, Espoo (FI)**

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1320 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: 11/881,367

(22) Filed: **Jul. 26, 2007**

(65) **Prior Publication Data**

US 2007/0291849 A1 Dec. 20, 2007

Related U.S. Application Data

(62) Division of application No. 10/421,629, filed on Apr. 23, 2003, now Pat. No. 7,263,125.

(60) Provisional application No. 60/374,667, filed on Apr. 23, 2002.

(51) Int. Cl.
H04N 7/12 (2006.01)

(52) U.S. Cl. 375/240.03; 375/240.16

(58) **Field of Classification Search** 375/240.04,
375/240.16; 341/107

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,054,103 A	10/1991	Yasuda et al.	382/56
5,144,426 A	9/1992	Tanaka et al.	358/133
5,231,484 A	7/1993	Gonzales et al.	358/133

5,237,410	A	8/1993	Inoue	358/136
5,715,009	A	2/1998	Tahara et al.	348/423
5,751,358	A	5/1998	Suzuki et al.	348/405
5,835,030	A	11/1998	Tsutsui et al.	341/51
5,907,362	A	5/1999	Yamamoto	348/405
6,256,349	B1	7/2001	Suzuki et al.	375/240.18
6,263,022	B1	7/2001	Chen et al.	375/240.0
6,677,868	B2 *	1/2004	Kerofsky et al.	341/107
6,879,632	B1	4/2005	Yokoyama	375/240
7,263,125	B2 *	8/2007	Iainema	375/240.04
7,367,041	B1	4/2008	Morishita et al.	725/37
2003/0152146	A1 *	8/2003	Lin et al.	375/240.16

FOREIGN PATENT DOCUMENTS

JP	06-121171		4/1994
JP	11-004449		1/1999
WO	WO 00 21302	A1	4/2000
WO	WO 01/26381	A1	4/2001

OTHER PUBLICATIONS

Single-pass constant-and variable-bit-rate MPEG-2 video compression, N. Mohsenian, R. Rajagopalan, C.A. Gonzales, Jul. 4, 1999, XP-002216512.

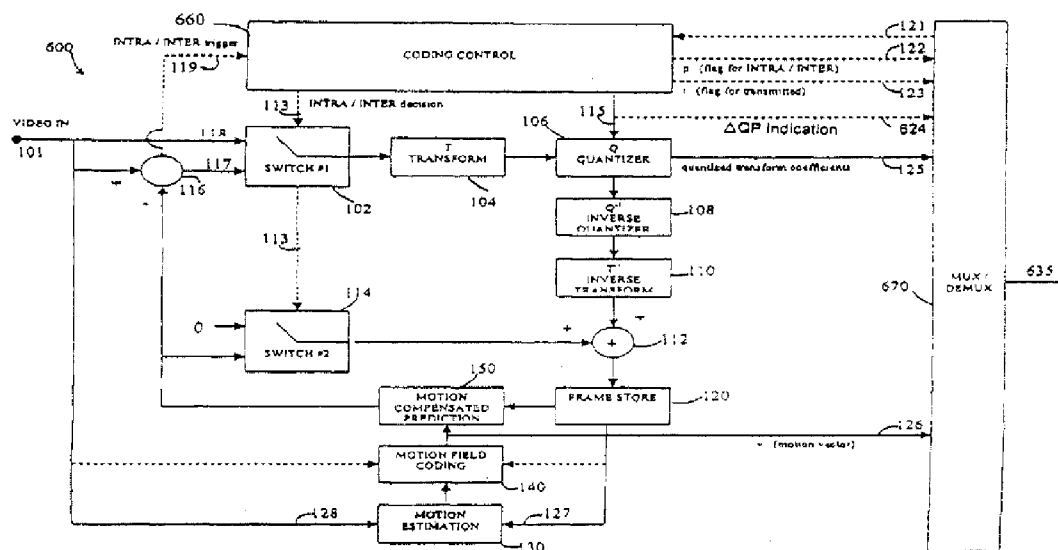
(Continued)

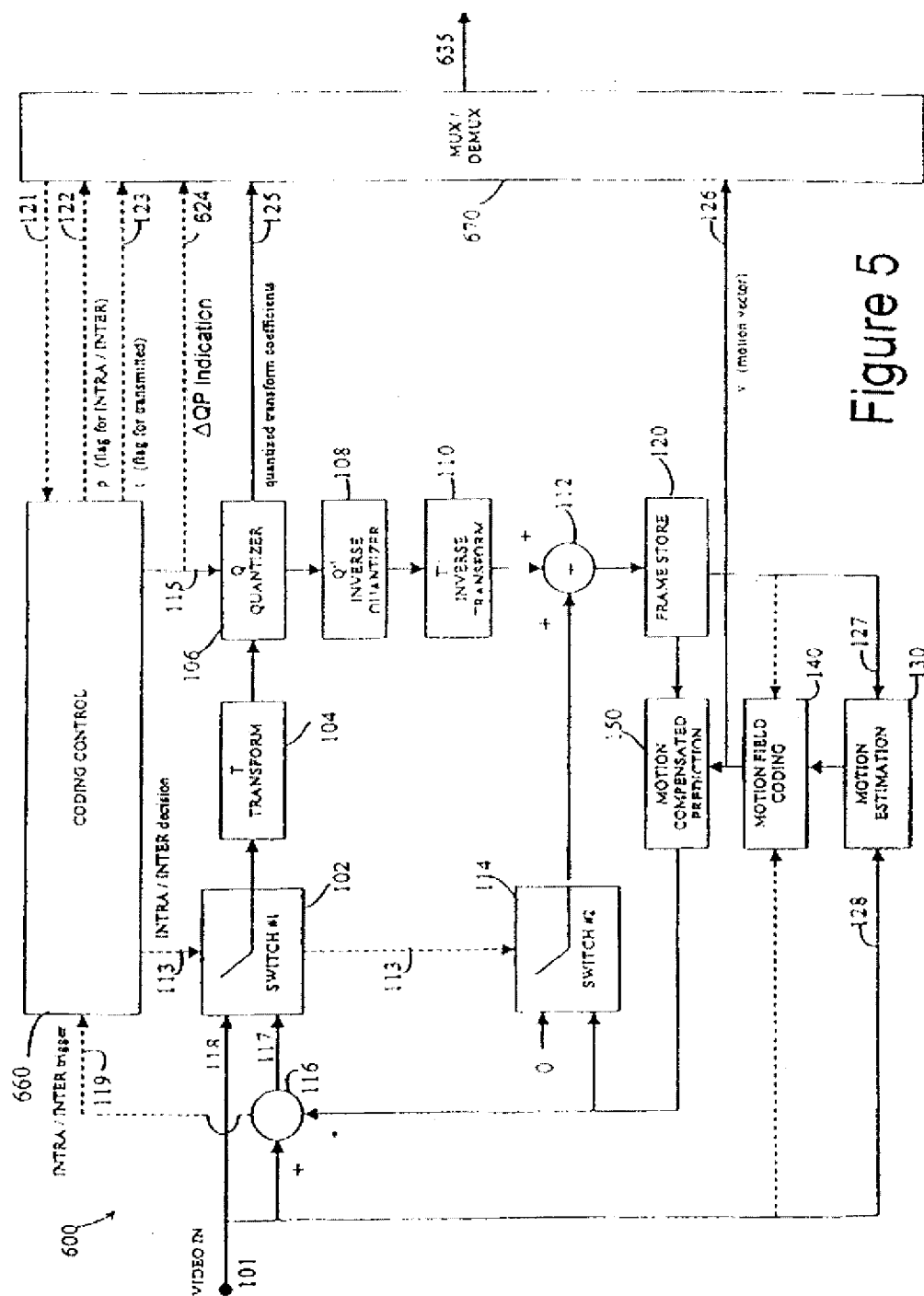
Primary Examiner — Gims Philippe

(57) **ABSTRACT**

A method and device for coding of digital video sequence, wherein an indication of quantization parameter (QP) is provided in the encoded bit-stream for decoding purposes. The QP related information is indicated by introducing a sequence level quantization parameter value SQP. More specifically, instead of coding the absolute values of picture/slice QPs, an indication of the difference ΔQP between the sequence level quantization parameter SQP and the picture/slice QP is provided. This eliminates the need to transmit a full QP for every picture/slice, and enables a statistically smaller difference value to be transmitted, thus providing a reduction in transmission bit-rate. The difference value is subsequently used in a corresponding decoder to reconstruct the picture/slice QP.

23 Claims, 8 Drawing Sheets





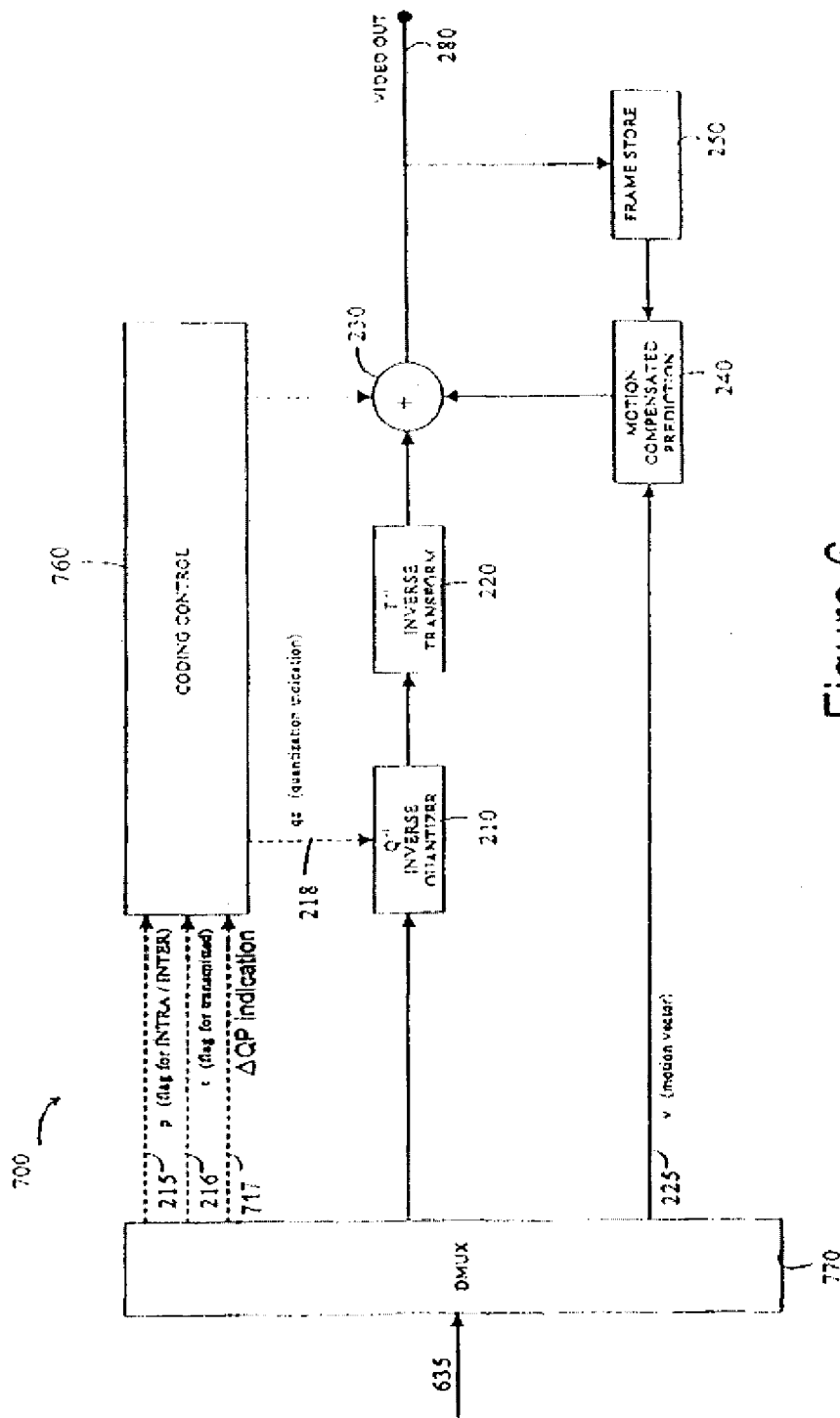


Figure 6

EXHIBIT 15



US008077991B2

(12) **United States Patent**
Lainema

(10) **Patent No.:** **US 8,077,991 B2**
(45) **Date of Patent:** **Dec. 13, 2011**

(54) **SPATIALLY ENHANCED TRANSFORM CODING**

(75) Inventor: **Jani Lainema**, Tampere (FI)

(73) Assignee: **Nokia Corporation**, Espoo (FI)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 915 days.

(21) Appl. No.: **12/101,019**

(22) Filed: **Apr. 10, 2008**

(65) **Prior Publication Data**

US 2008/0260270 A1 Oct. 23, 2008

Related U.S. Application Data

(60) Provisional application No. 60/911,480, filed on Apr. 12, 2007.

(51) **Int. Cl.**
G06K 9/46 (2006.01)

(52) **U.S. Cl.** **382/236**; 382/232; 382/238

(58) **Field of Classification Search** 382/238,
382/232, 166, 103, 173, 275; 348/169, 581;
704/500, 219; 375/240.12, 240.14, 244
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,125,861 A * 11/1978 Mounts et al. 375/240.12
5,598,214 A * 1/1997 Kondo et al. 375/240.14
6,498,810 B1 * 12/2002 Kim et al. 375/240
2007/0064797 A1 3/2007 Miao et al.

FOREIGN PATENT DOCUMENTS

JP 02002152049 A * 5/2002

OTHER PUBLICATIONS

Santiago P., et al. "Using Convex Set Techniques for Combined Pixel and Frequency Domaincoding of Time-Varying Images." IEEE Journal on Selected Areas in Communications, IEEE Service Center, vol. SAC-05, No. 7, Aug. 1, 1987, pp. 1127-1139.

Smith J., et al. "Joint adaptive space and frequency basis selection." IMAGE Processing, Proceedings., International Conference on Santa Barbara, CA, USA Oct. 26-29, 1997, vol. 3, pp. 702-705.

Clarke, R. J. "On Transform Coding Motion-Compensated Difference Images" IEE Proceedings I. Solid-State & Electron Devices, Institution of Electrical Engineers. vol. 139, No. 3 part 1, Jun. 1, 1992, pp. 372-376.

Wiegand T. et al. "Overview of the H.264/AVC video coding standard." IEEE Transactions on Circuits and Systems for Video Technology, IEEE Service Center, vol. 13, No. 7, Jul. 1, 2003, pp. 560-576.

Segall C. A. et al. "Pre- and post-processing algorithms for compressed video enhancements." Signals, Systems and Computers. Conference Record of the Thirty-Fourth Asilomar Conference on Oct. 29-Nov. 1, 2000, vol. 2, pp. 1369-1373.

The International Search Report PCT Application No. PCT/IB2008/051351.

English translation of Office action for corresponding Korean Application No. 2009-7023527 dated Jan. 6, 2011.

(Continued)

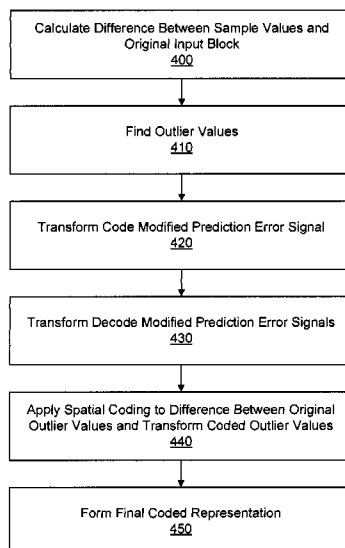
Primary Examiner — Anh Hong Do

(74) *Attorney, Agent, or Firm* — AlbertDhand LLP

(57) **ABSTRACT**

A system and method for improving prediction error coding performance of various video compression algorithms. Various embodiments combine the energy compaction features of transform coding with localization properties of spatial coding. In various embodiments, the effects of pixel "outliers" are removed from the transform and are coded separately as a spatially coded pixel prediction, thereby improving the coding gain of the transform.

39 Claims, 9 Drawing Sheets



OTHER PUBLICATIONS

Office Action for Chinese Patent Application No. 200880017526.3., dated May 18, 2011.

English translation of Office Action for Chinese Patent Application No. 200880017526.3., dated May 18, 2011.

Office Action for Mexican Patent Application No. MX/a/2009/010921, dated Mar. 23, 2011.

English translation of Office Action for Mexican Patent Application No. MX/a/2009/010921, dated Mar. 23, 2011.

* cited by examiner

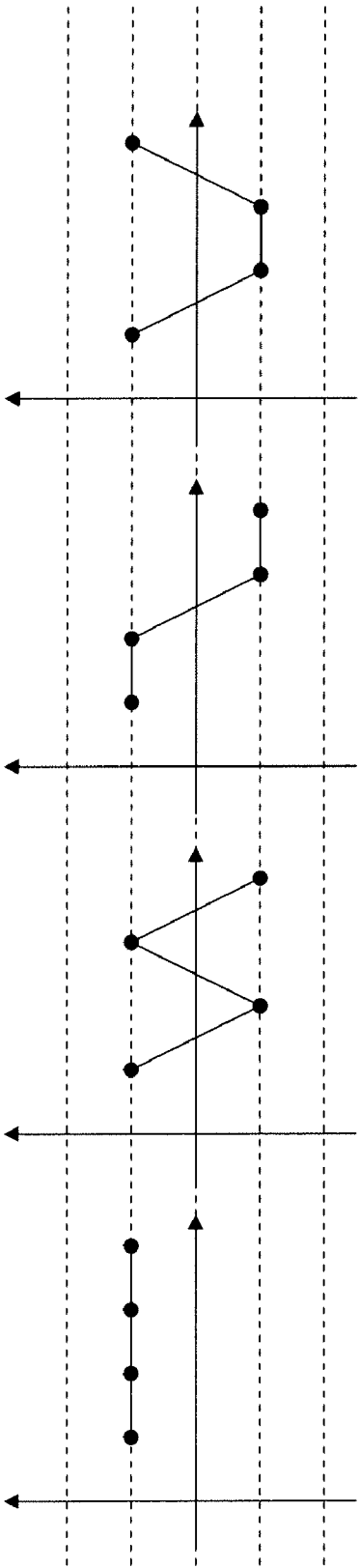


FIG. 1

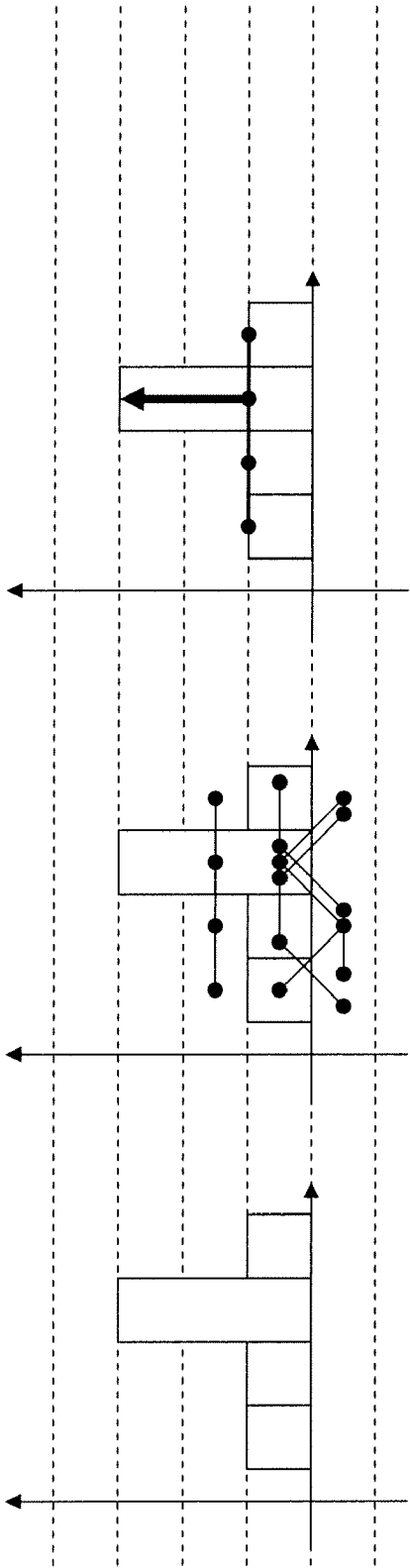


FIG. 2(a) FIG. 2(b) FIG. 2(c)

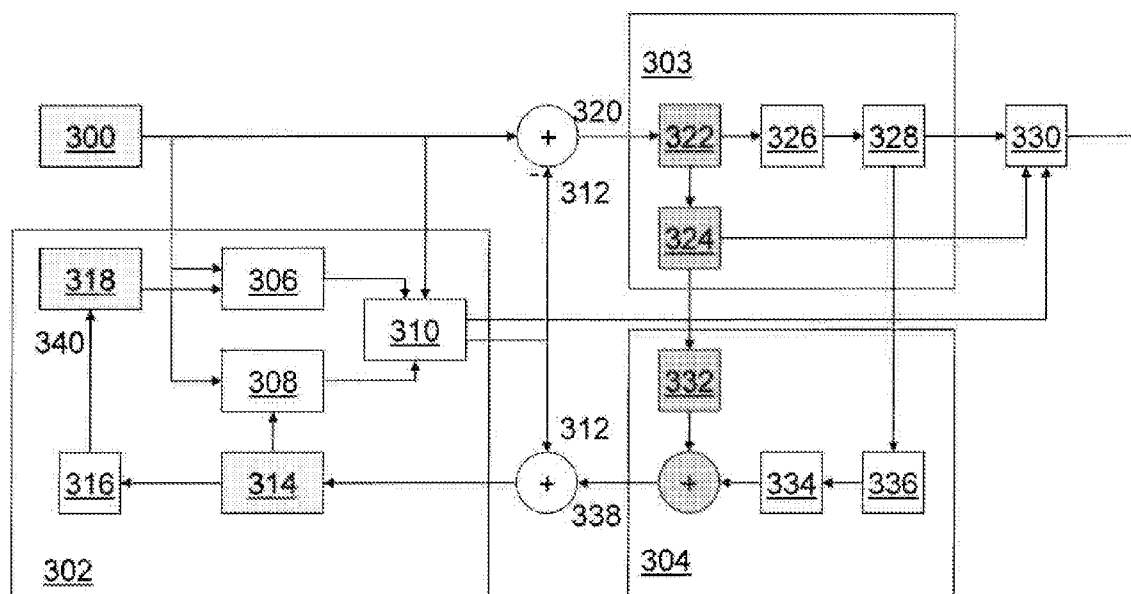


FIG. 3

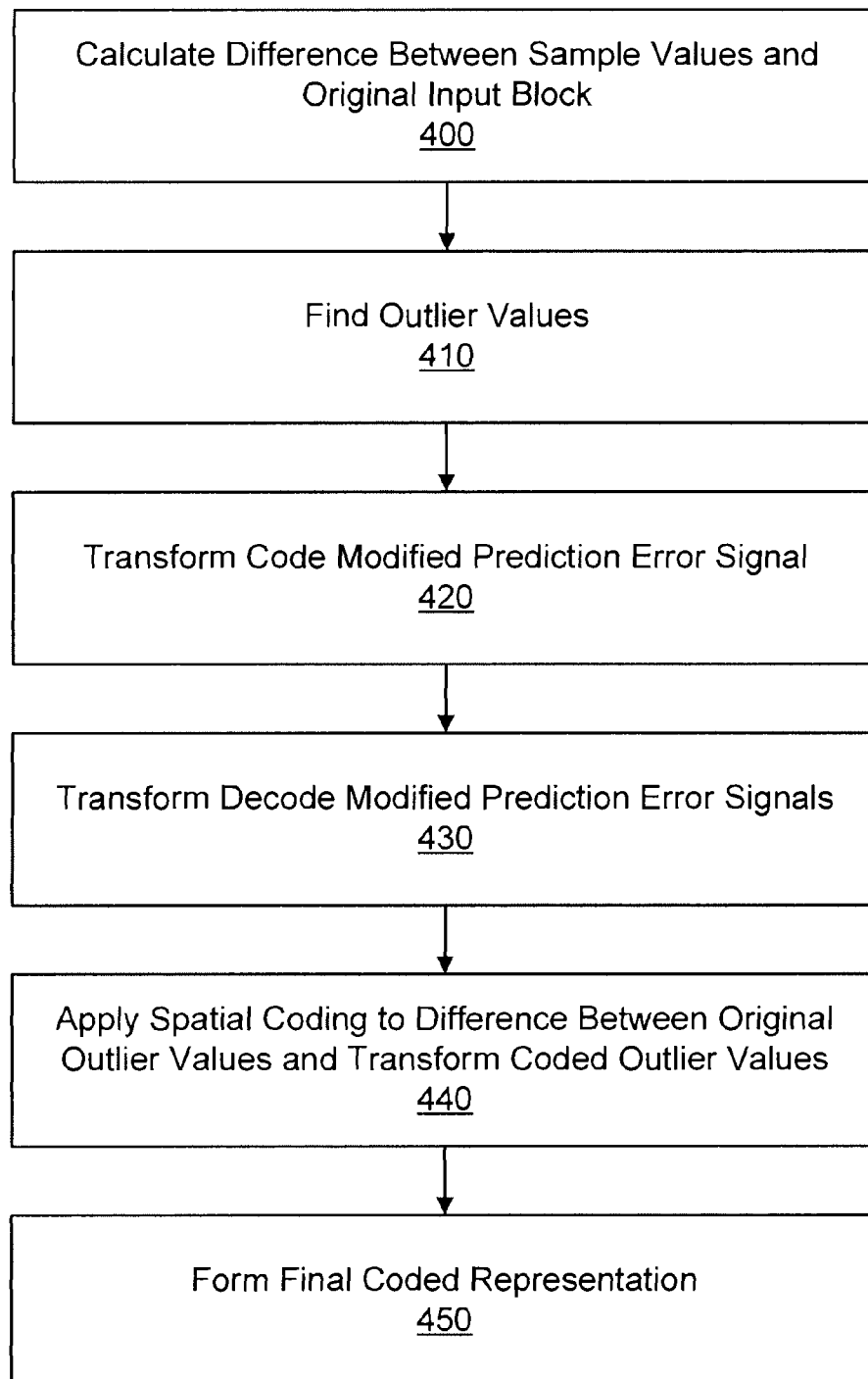


FIG. 4

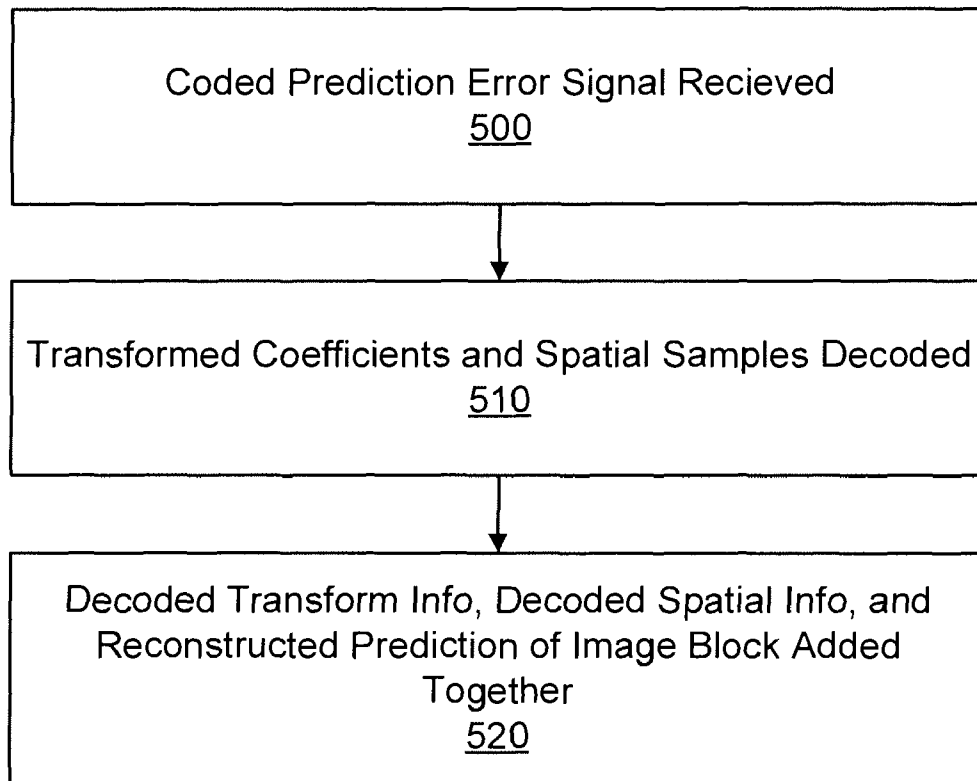


FIG. 5

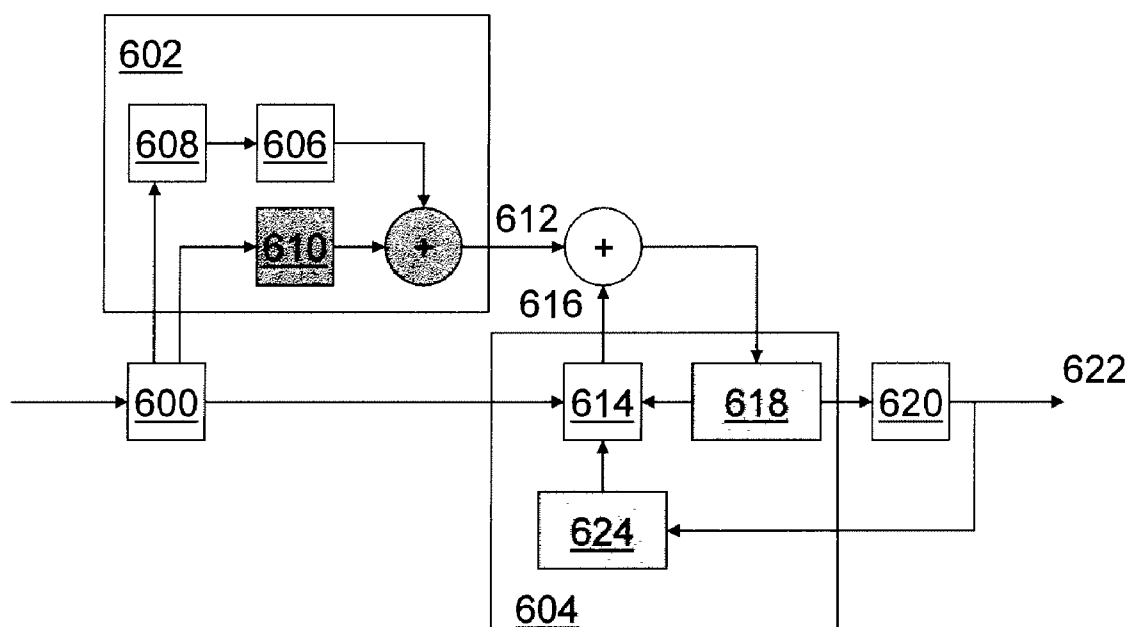
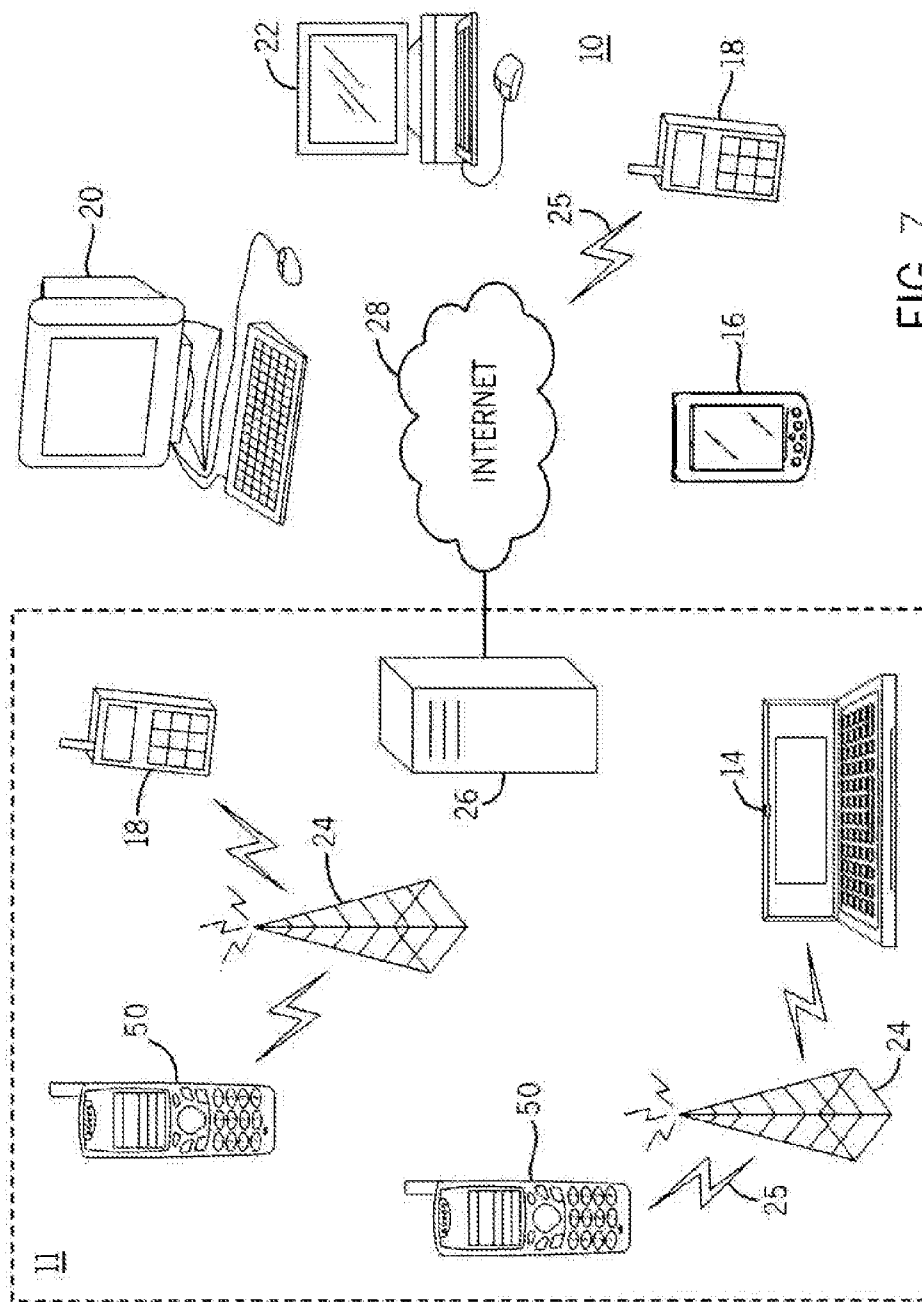


FIG. 6



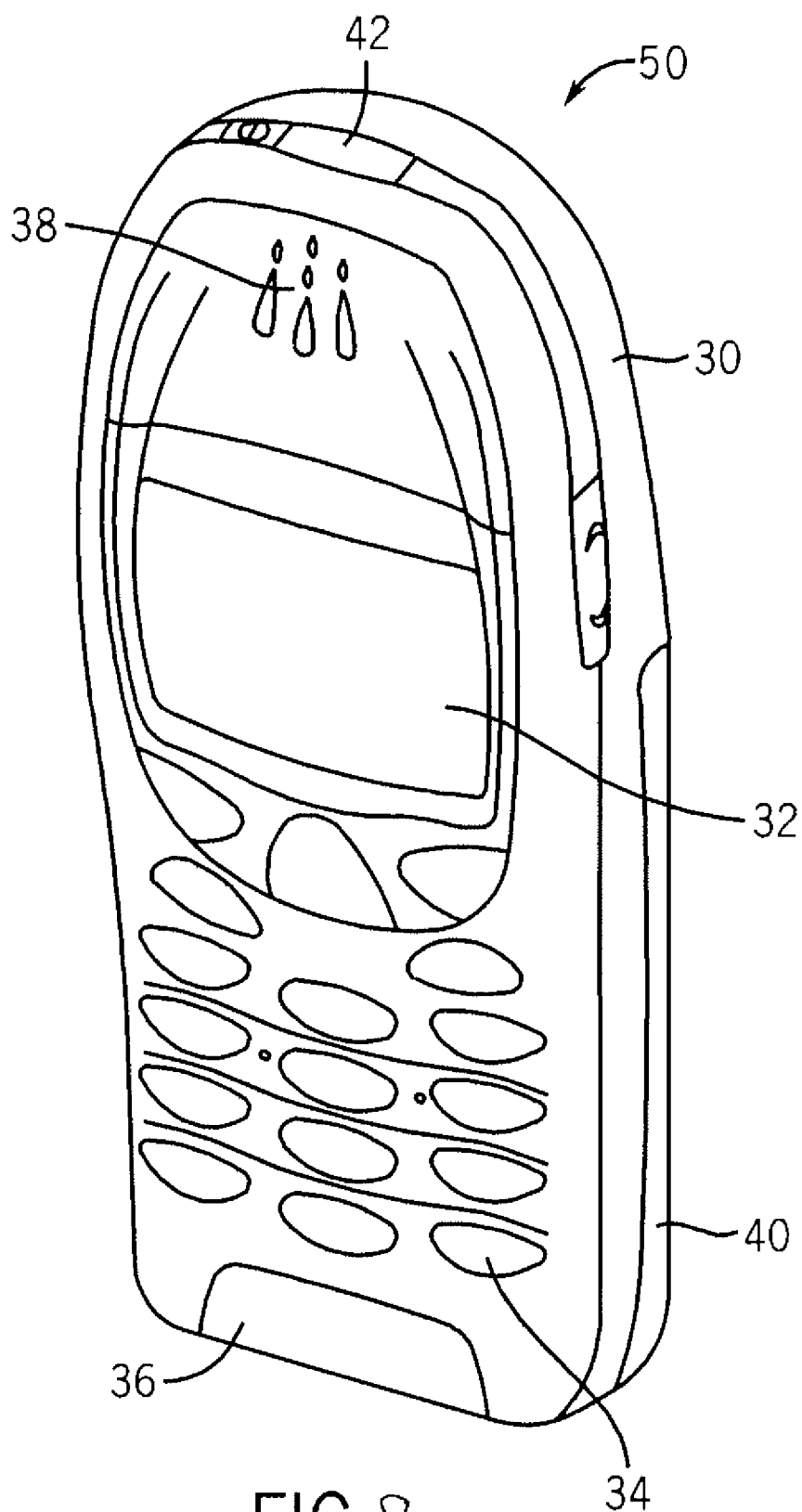


FIG. 8

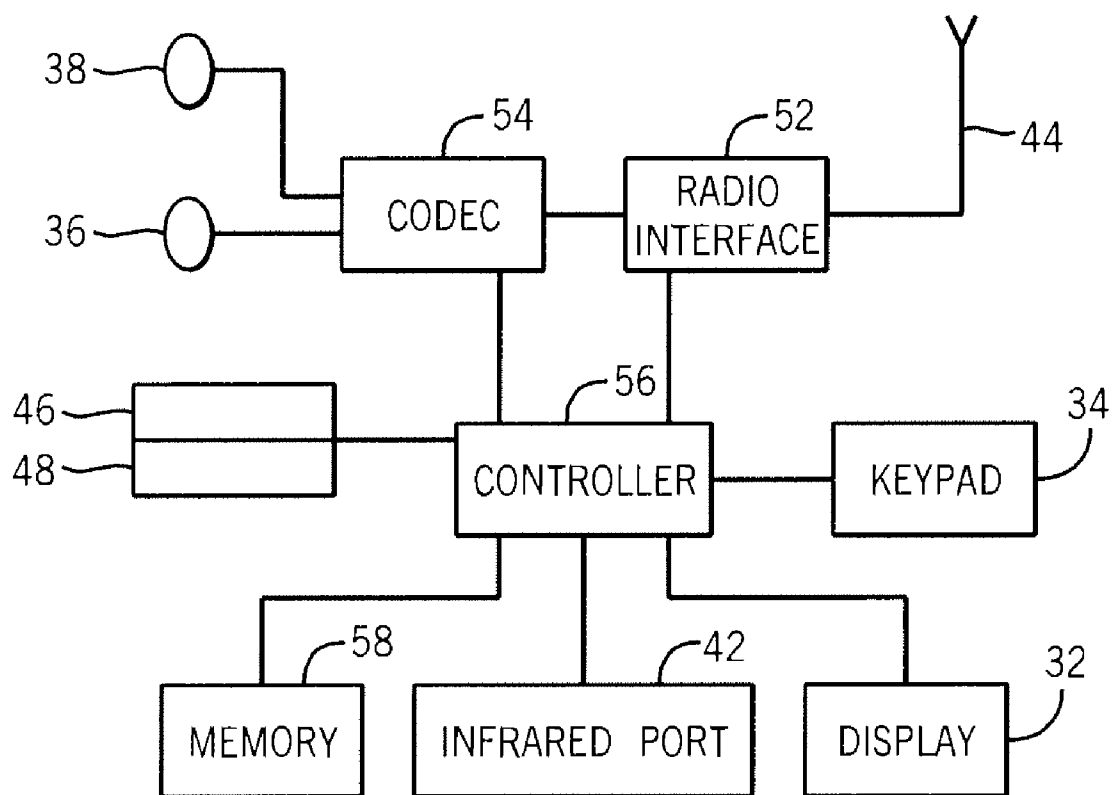


FIG. 9

1

SPATIALLY ENHANCED TRANSFORM CODING

FIELD OF THE INVENTION

The present invention relates to the coding and decoding of digital video material. More particularly, the present invention relates to prediction error coding in both frequency and spatial domains.

BACKGROUND OF THE INVENTION

This section is intended to provide a background or context to the invention that is recited in the claims. The description herein may include concepts that could be pursued, but are not necessarily ones that have been previously conceived or pursued. Therefore, unless otherwise indicated herein, what is described in this section is not prior art to the description and claims in this application and is not admitted to be prior art by inclusion in this section.

A video codec comprises an encoder that transforms input video into a compressed representation suited for storage and/or transmission and a decoder that can uncompress the compressed video representation back into a viewable form. Typically, the encoder discards some information in the original video sequence in order to represent the video in a more compact form, i.e., at a lower bitrate.

Typical hybrid video codecs, for example ITU-T H.263 and H.264, encode video information in two phases. In the first phase, pixel values in a certain picture area or "block" are predicted. These pixel values can be predicted, for example, by motion compensation mechanisms, which involve finding and indicating an area in one of the previously coded video frames that corresponds closely to the block being coded. Additionally, pixel values can be predicted via by spatial mechanisms, which involve using the pixel values around the block to be coded in a specified manner. The second phase involves coding the prediction error, i.e. the difference between the predicted block of pixels and the original block of pixels. This is typically accomplished by transforming the difference in pixel values using a specified transform (e.g., a Discrete Cosine Transform (DCT) or a variant thereof), quantizing the coefficients, and entropy coding the quantized coefficients. By varying the fidelity of the quantization process, the encoder can control the balance between the accuracy of the pixel representation (i.e., the picture quality) and the size of the resulting coded video representation (i.e., the file size or transmission bitrate).

The decoder reconstructs output video by applying prediction mechanisms that are similar to those used by the encoder in order to form a predicted representation of the pixel blocks (using motion or spatial information created by the encoder and stored in the compressed representation) and prediction error decoding (the inverse operation of the prediction error coding, recovering the quantized prediction error signal in the spatial pixel domain). After applying prediction and prediction error decoding processes, the decoder sums up the prediction and prediction error signals (i.e., the pixel values) to form the output video frame. The decoder (and encoder) can also apply additional filtering processes in order to improve the quality of the output video before passing it for display and/or storing it as a prediction reference for the forthcoming frames in the video sequence.

In typical video codecs, the motion information is indicated with motion vectors associated with each motion-compensated image block. Each of these motion vectors represents the displacement of the image block in the picture to be

2

coded (in the encoder side) or decoded (in the decoder side) and the prediction source block in one of the previously coded or decoded pictures. In order to represent motion vectors efficiently, motion vectors are typically coded differentially with respect to block specific predicted motion vectors. In a typical video codec, the predicted motion vectors are created in a predefined way, for example by calculating the median of the encoded or decoded motion vectors of the adjacent blocks.

Typical video encoders utilize Lagrangian cost functions to find optimal coding modes, e.g., the desired macroblock mode and associated motion vectors. This kind of cost function uses a weighting factor λ to tie together the exact or estimated image distortion due to lossy coding methods and the exact or estimated amount of information that is required to represent the pixel values in an image area:

$$C = D + \lambda R \quad (1)$$

In Eq. (1), C is the Lagrangian cost to be minimized, D is the image distortion (e.g., the mean squared error) with the mode and motion vectors considered, and R the number of bits needed to represent the required data to reconstruct the image block in the decoder (including the amount of data to represent the candidate motion vectors).

Transform coding of the prediction error signal in video or image compression system typically comprises DCT-based linear transform, quantization of the transformed DCT coefficients, and context based entropy coding of the quantized coefficients. However, the transform can efficiently pack energy of the prediction error signal only under certain statistics, and the coding performance deteriorates when the prediction error to be transformed becomes less correlated. This causes suboptimal performance, especially in modern video and image coding systems employing advanced motion compensation and spatial prediction processes in order to achieve good quality predictions for the image blocks to be coded (thus, minimizing and decorrelating the prediction error signal).

To address some of the above issues, a number of hybrid video coding schemes have been developed. These hybrid systems typically comprise a hybrid of two redundancy reduction techniques—prediction and transformation. Prediction can take the form of inter-picture prediction, which is used to remove temporal redundancies in the signal. Intra-picture prediction may also be used in the H.264/Advanced Video Coding (AVC) standard where spatial redundancies are removed by exploiting the similarities between neighboring regions within a picture frame. As a consequence of these inter-picture and intra-picture prediction techniques, a residual/error signal is formed by removing the predicted picture frame from the original. This prediction error signal is then typically block transform coded using an 8x8 DCT transform in order to reduce spatial redundancies in the signal.

SUMMARY OF THE INVENTION

Various embodiments of the present invention provide a system and method for representing the prediction error signal as a weighted sum of different basis functions of a selected transform and quantized spatial samples. The basis functions of the selected transform may comprise an orthogonal set of basis vectors, or the basis functions may not comprise an orthogonal set. According to various embodiments, the prediction error signal for a single image block is constructed using both transform basis functions and spatial samples (i.e., pixel values), thereby combining the desired features of both

3

the transform and spatial coding approaches discussed previously. This allows for the utilization of those selected transform basis functions that give good overall representation of the image block with minimal amount of transform coefficients (representing the component of prediction error signal that is well correlated with the basis functions). Additionally, various embodiments of the present invention allow for the efficient spatial representation of those components of the prediction error signal of the same image block that are not well correlated with the basis functions of the applied transform (such as certain types of sensor noise, high frequency texture and edge information).

According to various embodiments of the present invention, a system and method of encoding a prediction error signal for a block of data comprises calculating a difference signal representing differences between sample values of a predicted block of data and values for an original input block. Transform coding and spatial coding are both performed to the difference signal, thereby creating a first and second representations of first and second components of the difference signal. The first and second representations are then combined to provide the prediction error signal.

Various embodiments of the present invention also provide a system and method of decoding a prediction error signal for a block of data, comprising receiving a coded prediction error signal, the coded prediction error signal including a plurality of transform coefficients and a plurality of spatial samples. The plurality of transformed coefficients are decoded into decoded transform information, and the plurality of spatial samples are decoded into decoded spatial information. The decoded transform information, the decoded spatial information, and a reconstructed prediction of the block of data are then added, thereby forming a decoded representation of the block of data.

The implementations of various embodiments of the present invention serve to improve the compression efficiency of modern video and image codecs. Although a certain amount of increased computational complexity of encoding may be needed, fast algorithms can be applied in order to lower the encoding complexity that approaches the complexity level for traditional transform based coding. Any effect on the complexity for a decoder is negligible when implementing various embodiments of the present invention.

These and other advantages and features of the invention, together with the organization and manner of operation thereof, will become apparent from the following detailed description when taken in conjunction with the accompanying drawings, wherein like elements have like numerals throughout the several drawings described below.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows an example of basis functions of a Hadamard transform that can be used to encode image data;

FIG. 2(a) is an example of four scalar values that are to be coded or decoded and representing the prediction error signal between the original image and the prediction image; FIG. 2(b) shows the four weighted basis functions from FIG. 1 being summed to the prediction error signal; and FIG. 2(c) shows how the same prediction error signal can be reconstructed by summing up only the first basis function and a single spatial sample represented by an arrow;

FIG. 3 is a block diagram of a video encoder constructed in accordance with one embodiment of the present invention;

FIG. 4 is a flow chart showing one encoder algorithm which may be used in an embodiment of the present invention;

4

FIG. 5 is a flow chart showing one decoder algorithm which may be used in various embodiments of the present invention;

FIG. 6 is a block diagram of a video decoder constructed in accordance with one embodiment of the present invention;

FIG. 7 is an overview diagram of a system within which various embodiments of the present invention may be implemented;

FIG. 8 is a perspective view of an electronic device that can be used in conjunction with the implementation of various embodiments of the present invention; and

FIG. 9 is a schematic representation of the circuitry which may be included in the electronic device of FIG. 8.

DETAILED DESCRIPTION OF VARIOUS EMBODIMENTS

Various embodiments of the present invention provide a system and method for representing the prediction error signal as a weighted sum of different basis functions of a selected transform and quantized spatial samples. According to various embodiments, the prediction error signal for a single image block is constructed using both transform basis functions and spatial samples (i.e., pixel values), thereby combining the desired features of both the transform and spatial coding approaches discussed previously. This allows for the utilization of those selected transform basis functions that give good overall representation of the image block with minimal amount of transform coefficients (representing the component of prediction error signal that is well correlated with the basis functions). Additionally, various embodiments of the present invention allow for the efficient spatial representation of those components of the prediction error signal of the same image block that are not well correlated with the basis functions of the applied transform (such as certain types of sensor noise, high frequency texture and edge information).

FIGS. 1 and 2 show a simplified example of how various embodiments of the present invention are operable in one dimension. FIG. 1 shows an example of basis functions of a one-dimensional linear transform that can be used to encode image data (namely a Hadamard transform). The weighted sum of these four basis functions can be used to represent any four samples. FIG. 2(a) is an example of four scalar values that are to be coded or decoded and representing the prediction error signal between the original image and the prediction image. In FIG. 2(b), the four weighted basis functions from FIG. 1 are summed to the prediction error signal (1.5 times the first, 0.5 times the second, -0.5 times the third and -0.5 times the fourth basis function). In FIG. 2(c) it is demonstrated how the same prediction error signal can be reconstructed by summing up only the first basis function and a single spatial sample represented by an arrow. In this example, compression efficiency improvement can be expected, as the signal is represented by a single transform coefficient and a single spatial sample instead of four transform coefficients weighting the four Hadamard basis functions (as shown in FIG. 2(b)).

FIG. 3 is a block diagram of a video encoder constructed in accordance with one embodiment of the present invention. More particularly, FIG. 3 shows how an image to be encoded 300 undergoes pixel prediction 302, prediction error coding 303 and prediction error decoding 304. For pixel prediction 302, the image 300 undergoes both inter-prediction 306 and intra-prediction 308 which, after mode selection 310, results in prediction representation of an image block 312. A preliminary reconstructed image 314 is also used for intra-pre-

5

diction 308. Once all of the image blocks are processed, the preliminary reconstructed image 314 undergoes filtering at 316 to create a final reconstructed image 340, which is sent to a reference frame memory 318 and is also used for inter-prediction 306 of future frames.

The prediction representation of the image block 312, as well as the image to be encoded 300, are used together to define a prediction error signal 320 which is used for prediction error coding 303. In prediction error coding 303, the prediction error signal 320 undergoes feature selection 322 and spatial quantization 324, as well as transform 326 and quantization 328 (both after feature selection 322). The data describing prediction error and predicted representation of the image block 312 (e.g., motion vectors, mode information and quantized DCT+spatial samples) are passed to entropy coding 330. The prediction error decoding 304 is substantially the opposite of the prediction error coding 303, with the prediction error decoding including an inverse spatial quantizer 332, an inverse transform 334 and an inverse quantization 336. The result of the prediction error decoding 304 is a reconstructed prediction error signal 338, which is used in combination with the predicted representation of the image block 312 to create the preliminary reconstructed image 314.

FIG. 4 is a flow chart showing one encoder algorithm which may be used in an embodiment of the present invention. However, as discussed below, a variety of different algorithms may be used according to principles of the present invention. At 400 in FIG. 4, the difference between sample values of a predicted block and an original input block is calculated. It should be noted that "values" and "outlier values" as used herein are intended to refer to these difference values. At 410, outlier values are found. In this case, the amplitude depends on the expected accuracy of the coded representation; smaller amplitude representations qualify as outliers if one is targeting high bitrate good quality representation, and only high amplitude representations qualify if one is targeting a lower bitrate, lower quality representation. At 420, the modified prediction error signal is transform coded (involving transform, quantization and entropy coding), with the outlier values being substituted by interpolated representations of those values, e.g., by averaging the neighboring prediction error values. At 430, the modified prediction error signals are transform decoded back to the spatial domain. At 440, spatial coding is applied (involving quantization and entropy coding without transform) to the difference between the original outlier values and the transform decoded outlier values. At 450, the final coded representation of the prediction error signal is formed by joining the transform coded data and the spatial coded data. In an alternative embodiment, the spatial coding occurs before transform coding.

When considering the one-dimensional example depicted in FIGS. 2(a)-2(c), the algorithm outlined in FIG. 4 would process the relevant data as follows. As discussed previously, FIG. 2(a) represents the difference signal [1 1 3 1] to be coded. In this case, [3] is an outlier. Therefore, the prediction error signal is modified by substituting the [3] with an average of the second and fourth samples, making the signal [1 1 1 1]. The difference between the original outlier sample ([3]) and the transform coded outlier value ([1]) is then spatial coded, making the spatial coded signal [0 0 2 0]. The entropy coded representations of the transform coded and spatial coded signals are then written to the bitstream. Thereafter, the decoder can recover the coded signal by adding up the inverse transformed data [1 1 1 1] and the inverse spatial coded data [0 0 2 0] (together with the prediction samples of the image block).

When considering the above example, it is important to note that the choice of Hadamard basis vectors for the trans-

6

form is only intended to be exemplary in nature. In fact, the various methods and techniques described herein can be applied to any transform employing basis functions, and the basis functions do not necessarily need to be orthogonal.

A video or image codec, according to various embodiments the present invention, can be implemented in various ways. In the case of an encoder, the encoder can use different strategies to find the desired transform coefficients and spatial samples. For example, an encoder can first transform code the signal and apply spatial coding to the residual. The encoder can also first apply spatial coding to the signal, followed by transform coding of the residual. Additionally, some or all of the transform coefficients can be set to zero in order to improve the coding performance. Some or all spatial samples can also be set to zero in order to improve the coding performance. An encoder can also iteratively modify transform coefficients and/or spatial samples in order to improve the coding performance until a desired performance or a maximum defined number of iterations is achieved.

In terms of quantization and dequantization, the quantization and dequantization of transform coefficients and spatial samples can be tied together (e.g., a quantization step size for both transform coefficients and spatial samples can be derived from a single parameter). Alternatively, different quantizations and dequantizations can be applied to transform coefficients and spatial samples.

For the coding and decoding of spatial samples, such coding and decoding can depend on transform coefficients and vice versa. Alternatively, the coding and decoding of spatial samples, transform coefficients or both can depend on the prediction signal, on the other transform coefficients and spatial samples in the same image or in other images.

In addition to the above, it can be indicated that there are no transform coefficients or spatial samples for a specific image area or image. It can also be indicated that only spatial coding or only transform coding is used for a specific image area or image. The number of spatial samples can be coded and decoded as one unit, for example representing certain patterns of textures. Pre- and/or post-processing mechanisms can be applied to the prediction signal, prediction error signal reconstructed signal or any combination thereof. The method can be used to code and decode other information instead or in addition to the prediction error signals. The codec can limit the usage of either transform coefficients (e.g., allow only low frequency transform coefficients to be present in the coded representation of the signal) or spatial samples.

FIG. 5 shows a decoding process according to various embodiments of the present invention. At 500 in FIG. 5, a video decoder receives a coded prediction error signal which comprises both transformed coefficients and spatial samples. At 510, both the transformed coefficients and the spatial samples are decoded. At 520, the decoded transform information, the decoded spatial information and reconstructed prediction of the image block are added together to form decoded representation of the image block.

FIG. 6 is a block diagram of a video decoder constructed in accordance with one embodiment of the present invention. As shown in FIG. 6, entropy decoding 600 is followed by both prediction error decoding 602 and pixel prediction 604. In prediction error decoding 602, in addition to an inverse transform 606 and inverse quantization 608, an inverse spatial quantizer 610 is used as discussed herein, ultimately resulting in a reconstructed prediction error signal 612. For pixel prediction 604, either intra-prediction or inter-prediction occurs at 614 to create a predicted representation of an image block 616. The predicted representation of the image block 616 is used in conjunction with the reconstructed prediction error

7

signal **612** to create a preliminary reconstructed image **618**, which in turn can be used both for prediction **614**. Once all of the image blocks have been processed, the preliminary reconstructed image **618** is passed for filtering **620**. The filtered image can also be stored in reference frame memory **624**, making it usable for prediction **614** as well.

FIG. 7 shows a system **10** in which various embodiments of the present invention can be utilized, comprising multiple communication devices that can communicate through one or more networks. The system **10** may comprise any combination of wired or wireless networks including, but not limited to, a mobile telephone network, a wireless Local Area Network (LAN), a Bluetooth personal area network, an Ethernet LAN, a token ring LAN, a wide area network, the Internet, etc. The system **10** may include both wired and wireless communication devices.

For exemplification, the system **10** shown in FIG. 7 includes a mobile telephone network **11** and the Internet **28**. Connectivity to the Internet **28** may include, but is not limited to, long range wireless connections, short range wireless connections, and various wired connections including, but not limited to, telephone lines, cable lines, power lines, and the like.

The exemplary communication devices of the system **10** may include, but are not limited to, an electronic device **50**, a combination personal digital assistant (PDA) **14** and mobile telephone **14**, a PDA **16**, an integrated messaging device (IMD) **18**, a desktop computer **20**, a notebook computer **22**, etc. The communication devices may be stationary or mobile as when carried by an individual who is moving. The communication devices may also be located in a mode of transportation including, but not limited to, an automobile, a truck, a taxi, a bus, a train, a boat, an airplane, a bicycle, a motorcycle, etc. Some or all of the communication devices may send and receive calls and messages and communicate with service providers through a wireless connection **25** to a base station **24**. The base station **24** may be connected to a network server **26** that allows communication between the mobile telephone network **11** and the Internet **28**. The system **10** may include additional communication devices and communication devices of different types.

The communication devices may communicate using various transmission technologies including, but not limited to, Code Division Multiple Access (CDMA), Global System for Mobile Communications (GSM), Universal Mobile Telecommunications System (UMTS), Time Division Multiple Access (TDMA), Frequency Division Multiple Access (FDMA), Transmission Control Protocol/Internet Protocol (TCP/IP), Short Messaging Service (SMS), Multimedia Messaging Service (MMS), e-mail, Instant Messaging Service (IMS), Bluetooth, IEEE 802.11, etc. A communication device involved in implementing various embodiments of the present invention may communicate using various media including, but not limited to, radio, infrared, laser, cable connection, and the like.

FIGS. 8 and 9 show one representative electronic device **50** within which the present invention may be implemented. It should be understood, however, that the present invention is not intended to be limited to one particular type of device. The electronic device **50** of FIGS. 8 and 9 includes a housing **30**, a display **32** in the form of a liquid crystal display, a keypad **34**, a microphone **36**, an ear-piece **38**, a battery **40**, an infrared port **42**, an antenna **44**, a smart card **46** in the form of a UICC according to one embodiment of the invention, a card reader **48**, radio interface circuitry **52**, codec circuitry **54**, a control-

8

ler **56** and a memory **58**. Individual circuits and elements are all of a type well known in the art, for example in the Nokia range of mobile telephones.

The various embodiments of the present invention described herein is described in the general context of method steps or processes, which may be implemented in one embodiment by a computer program product, embodied in a computer-readable medium, including computer-executable instructions, such as program code, executed by computers in networked environments. Generally, program modules may include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Computer-executable instructions, associated data structures, and program modules represent examples of program code for executing steps of the methods disclosed herein. The particular sequence of such executable instructions or associated data structures represents examples of corresponding acts for implementing the functions described in such steps or processes.

Software and web implementations of various embodiments of the present invention can be accomplished with standard programming techniques with rule-based logic and other logic to accomplish various database searching steps or processes, correlation steps or processes, comparison steps or processes and decision steps or processes. It should be noted that the words "component" and "module," as used herein and in the following claims, is intended to encompass implementations using one or more lines of software code, and/or hardware implementations, and/or equipment for receiving manual inputs.

The foregoing description of embodiments of the present invention have been presented for purposes of illustration and description. The foregoing description is not intended to be exhaustive or to limit embodiments of the present invention to the precise form disclosed, and modifications and variations are possible in light of the above teachings or may be acquired from practice of various embodiments of the present invention. The embodiments discussed herein were chosen and described in order to explain the principles and the nature of various embodiments of the present invention and its practical application to enable one skilled in the art to utilize the present invention in various embodiments and with various modifications as are suited to the particular use contemplated.

What is claimed is:

1. A method, comprising:

calculating, by a processor, a difference signal representing differences between sample values of a predicted block of data and values for an original input block;

performing, by a processor, transform coding to the difference signal, thereby creating a first representation of a first component of the difference signal;

performing, by a processor, spatial coding to the difference signal, thereby creating a second representation of a first component of the difference signal;

joining, by a processor, the first and second representations to form a coded prediction error signal; and
at least one of transmitting or storing the coded prediction error signal.

2. The method of claim 1, further comprising, before performing transform coding and spatial coding, substituting outlier values in the difference signal with non-outlier values to create a modified prediction error signal.

3. The method of claim 2, wherein transform coding is applied to the modified prediction error signal and spatial coding is applied to a residual thereof.

9

4. The method of claim 2, wherein spatial coding is applied to the modified prediction error signal and transform coding is applied to a residual thereof.

5. The method of claim 1, wherein, for transform coding, at least one transform coefficient is set to zero.

6. The method of claim 1, wherein, for spatial coding, at least one spatial sample is set to zero.

7. The method of claim 2, wherein the non-outlier values used to replace the outlier values comprise an average of neighboring prediction error values.

8. The method of claim 1, further comprising providing an indication that no transform coefficients exist in the block of data if appropriate.

9. The method of claim 1, further comprising providing an indication that no spatial samples exist in the block of data if appropriate.

10. The method of claim 1, wherein for the transform coding, a transform coefficient is formed, the transform coefficient comprising a weighting coefficient of a discrete transform basis vector.

11. The method of claim 10, wherein the discrete transform comprises a discrete orthogonal transform.

12. A computer program product, embodied in a non-transitory computer-readable medium, comprising computer code configured to perform the method of claim 1.

13. An apparatus, comprising:

a processor; and

a memory unit communicatively connected to a controller and comprising:

computer code for performing transform coding to the difference signal, thereby creating a first representation of a first component of the difference signal;

computer code for performing spatial coding to the difference signal, thereby creating a second representation of a first component of the difference signal; and
computer code for joining the first and second representations.

14. The apparatus of claim 13, wherein the memory unit further comprises computer code for, before performing transform coding and spatial coding, substituting outlier values in the difference signal with non-outlier values to create a modified prediction error signal.

15. The apparatus of claim 14, wherein transform coding is applied to the modified prediction error signal and spatial coding is applied to a residual thereof.

16. The apparatus of claim 14, wherein spatial coding is applied to the modified prediction error signal and transform coding is applied to a residual thereof.

17. The apparatus of claim 13, wherein, for transform coding, at least one transform coefficient is set to zero.

18. The apparatus of claim 13, wherein, for spatial coding, at least one spatial sample is set to zero.

19. The apparatus of claim 13, wherein the non-outlier values used to replace the outlier values comprise an average of neighboring prediction error values.

20. The apparatus of claim 13, further comprising providing an indication that no transform coefficients exist in the block of data if appropriate.

21. The apparatus of claim 13, further comprising providing an indication that no spatial samples exist in the block of data if appropriate.

22. A method, comprising:

receiving a coded prediction error signal, the coded prediction error signal including a plurality of transform coefficients and a plurality of spatial samples;

decoding, by a processor, the plurality of transformed coefficients into decoded transform information;

10

decoding, by a processor, the plurality of spatial samples into decoded spatial information; and

adding, by a processor, the decoded transform information, the decoded spatial information, and a reconstructed prediction of the block of data, thereby forming a decoded representation of the block of data.

23. The method of claim 22, wherein an indication is provided that no transform coefficients exist in the block of data if appropriate.

24. The method of claim 22, wherein an indication is provided that no spatial samples exist in the block of data if appropriate.

25. The method of claim 22, wherein the decoding of the plurality of transformed coefficients and spatial samples includes dequantization of each, and wherein the dequantization of transformed coefficients and spatial samples are related to each other.

26. The method of claim 22, wherein the decoding of the plurality of transformed coefficients and spatial samples includes dequantization of each, and wherein the dequantization of transformed coefficients and spatial samples are different from each other.

27. The method of claim 22, wherein the decoding of the spatial samples is dependent upon the decoding of the transformed coefficients.

28. The method of claim 22, wherein the decoding of the transformed coefficients is dependent upon the decoding of the spatial samples.

29. The method of claim 22, wherein at least some of the plurality of spatial samples are decoded as a single unit.

30. A computer program product, embodied in a non-transitory computer-readable medium, comprising computer code configured to perform the method of claim 22.

31. An apparatus, comprising:

a processor; and

a memory unit communicatively connected to the processor and including:

computer code for processing a received coded prediction error signal, the coded prediction error signal including a plurality of transformed coefficients and a plurality of spatial samples;

computer code for decoding the plurality of transformed coefficients into decoded transform information;

computer code for decoding the plurality of spatial samples into decoded spatial information; and

computer code for adding the decoded transform information, the decoded spatial information, and a reconstructed prediction of the block of data, thereby forming a decoded representation of the block of data.

32. The apparatus of claim 31, wherein an indication is provided that no transform coefficients exist in the block of data if appropriate.

33. The apparatus of claim 31, wherein an indication is provided that no spatial samples exist in the block of data if appropriate.

34. The apparatus of claim 31, wherein the decoding of the plurality of transformed coefficients and spatial samples includes dequantization of each, and wherein the dequantization of transformed coefficients and spatial samples are related to each other.

35. The apparatus of claim 31, wherein the decoding of the plurality of transformed coefficients and spatial samples includes dequantization of each, and wherein the dequantization of transformed coefficients and spatial samples are different from each other.

11

36. The apparatus of claim 31, wherein the decoding of the spatial samples is dependent upon the decoding of the transformed coefficients.

37. The apparatus of claim 31, wherein the decoding of the transformed coefficients is dependent upon the decoding of the spatial samples. 5

38. The apparatus of claim 31, wherein at least some of the plurality of spatial samples are decoded as a single unit.

39. A method, comprising: 10

calculating, by a processor, a difference signal representing differences between sample values of a predicted block of data and values for an original input block;

substituting, by a processor, outlier values in the difference signal with non-outlier values to create a modified prediction error signal; 15

12

performing, by a processor, one of transform coding and spatial coding to the modified prediction error signal, creating one of a transform coded data and spatial coded data

performing, by a processor, an inverse of the one of transform coding and spatial coding;

applying, by a processor, the other of transform coding and spatial coding to a residual from the inverse transform or spatial decoding, creating the other of transform coded data and spatial coded data;

joining, by a processor, the transform coded data and the spatial coded data to form a coded prediction error signal; and

at least one of transmitting or storing the coded prediction error signal.

* * * * *

EXHIBIT 16

- (51) **Int. Cl.**
H04N 19/51 (2014.01)
H04N 19/52 (2014.01)
- (58) **Field of Classification Search**
 USPC 375/240, 240.01–240.29
 See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2011/0170602	A1	7/2011	Lee et al.	
2011/0176013	A1 *	7/2011	Robertson	H04N 5/145 348/208.4
2011/0182362	A1	7/2011	Kim et al.	
2012/0230408	A1 *	9/2012	Zhou	H04N 19/105 375/240.15
2012/0257678	A1 *	10/2012	Zhou	H04N 19/436 375/240.16
2012/0269270	A1 *	10/2012	Chen	H04N 19/597 375/240.16
2012/0300846	A1 *	11/2012	Sugio	H04N 19/521 375/240.16
2012/0307905	A1	12/2012	Kim et al.	
2012/0320984	A1 *	12/2012	Zhou	H04N 19/50 375/240.16
2013/0003850	A1 *	1/2013	Sugio	H04N 19/105 375/240.16
2013/0004092	A1 *	1/2013	Sasai	H04N 19/70 382/233
2013/0070855	A1 *	3/2013	Zheng	H04N 19/105 375/240.16
2013/0083853	A1 *	4/2013	Coban	H04N 19/563 375/240.16
2013/0272408	A1 *	10/2013	Chen	H04N 19/597 375/240.16

OTHER PUBLICATIONS

Tai, Shen-Chuan, et al. "A multi-pass true motion estimation scheme with motion vector propagation for frame rate up-conversion applications." *Journal of display technology* 4.2 (2008): 188-197.*

Han, Woo-Jin, et al. "Improved video compression efficiency through flexible unit representation and corresponding extension of coding tools." *Circuits and Systems for Video Technology*, IEEE Transactions on 20.12 (2010): 1709-1720. (Previously Presented But with additional annotations).*

Tai, Shen-Chuan, et al. "A multi-pass true motion estimation scheme with motion vector propagation for frame rate up-conversion applications." *Journal of display technology* 4.2 (2008): 188-197. (Previously Presented But with additional annotations).*

Huang, Ai-Mei, and Truong Q. Nguyen. "A multistage motion vector processing method for motion-compensated frame interpolation." *Image Processing*, IEEE Transactions on 17.5 (2008): 694-708.*

International Search Report and Written Opinion received for corresponding Patent Cooperation Treaty Application No. PCT/FI2012/051070, dated Mar. 27, 2013, 12 pages.

Oudin, S. et al.; "Block Merging for Quadtree-Based Video Coding", IEEE Int. Conf. on Multimedia and Expo, Jul. 11-15, 2011, 6 pages.

Sullivan, G. J.; "Overview of the High Efficiency Video Coding (HEVC) Standard", IEEE Trans. on Circuits and Systems for Video Technology, vol. 22, No. 12, Dec. 2012, pp. 1649-1668.

Taiwanese Office Action and Search Report from Taiwanese Patent Application No. 101140777 dated Dec. 2, 2015.

Office Action from corresponding Korean Patent Application No. 2014-7015093, dated Aug. 21, 2015.

Wiegand, Tomas, et al.; "WD3: Working Draft 3 of High-Efficiency Video Coding"; Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11; 5th Meeting; Geneva, CH, Mar. 16-23, 2011; Document JCTVC-F603; 239 pages.

Bross, Benjamin, et al.; "WD4: Working Draft 4 of High-Efficiency Video Coding"; Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11; 6th Meeting; Torino, IT, Jul. 14-22, 2011; Document JCTVC-F803_d; 232 pages.

J.-L. Lin, Y.-W. Chen, Y.-W. Huang, S. Lei; "CE9: Results of Experiment ROB04"; JCT-VC Doc. JCTVC-F052; Jul. 2011.

Office Action from corresponding Canadian Patent Application No. 2,854,495, dated Oct. 7, 2015.

Extended European Search Report for corresponding European Application No. 12845839.5 dated Mar. 21, 2016, 10 pages.

Nakamura, H. et al., *Unification of derivation process for merge mode and MVP*, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 6th Meeting; Torino, IT, Jul. 14-22, 2011; Document JCTVC-F419; URL: http://wftp3.itu.int/AV-ARCH/JCTVC-SITE/2011_07_F_Torino/, 10 pages.

Zheng, Y. et al., *Merge Candidate Selection in 2Nx33 N, Nx33 2N, and Nx33 N Mode*, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 6th Meeting; Torino, IT, Jul. 14-22, 2011; Document JCTVC-F302, 6 pages.

Bross, B. et al., *Core Experiment 9: MV Coding and Skip/Merge Operations*, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 6th Meeting; Torino, IT, Jul. 14-22, 2011; Document JCTVC-F909, 13 pages.

Jeon, Y. et al., *On MVP list pruning process*, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 6th 2011, Meeting; Torino, IT, Jul. 14-22, 2011, Document JCTVC-F105, 7 pages.

Bici, O. et al., *Non-CE13: Simplification of merge mode*, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 7th Meeting; Geneva, CH, Nov. 21-30, 2011, Document JCTVC-G593; URL: <http://wftp3.itu.int/av-arch/jctvc-site/>, 13 pages.

Office Action from Korean Patent Application No. 2014-7015093 dated Aug. 22, 2016.

Office Action for Canadian Application No. 2,854,495 dated Sep. 6, 2016.

Office Action for Chinese Application No. 2012800657775 dated Oct. 9, 2016.

* cited by examiner

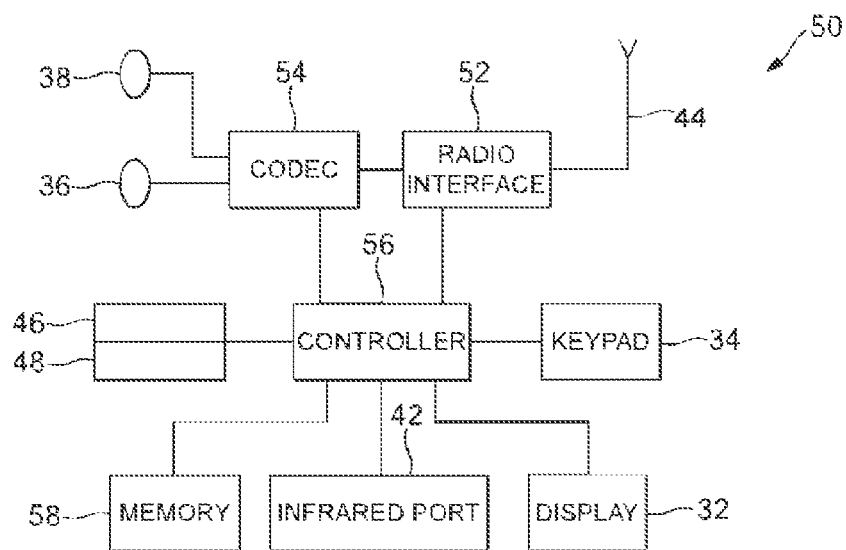


FIG. 1

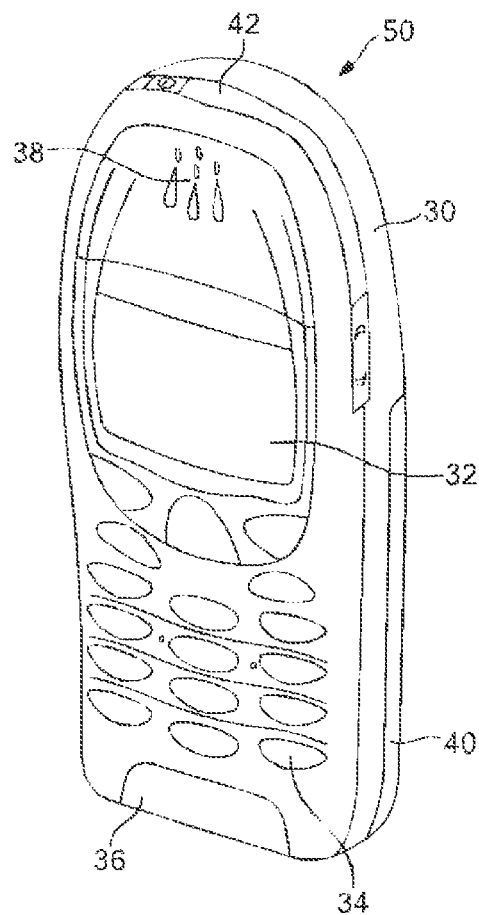
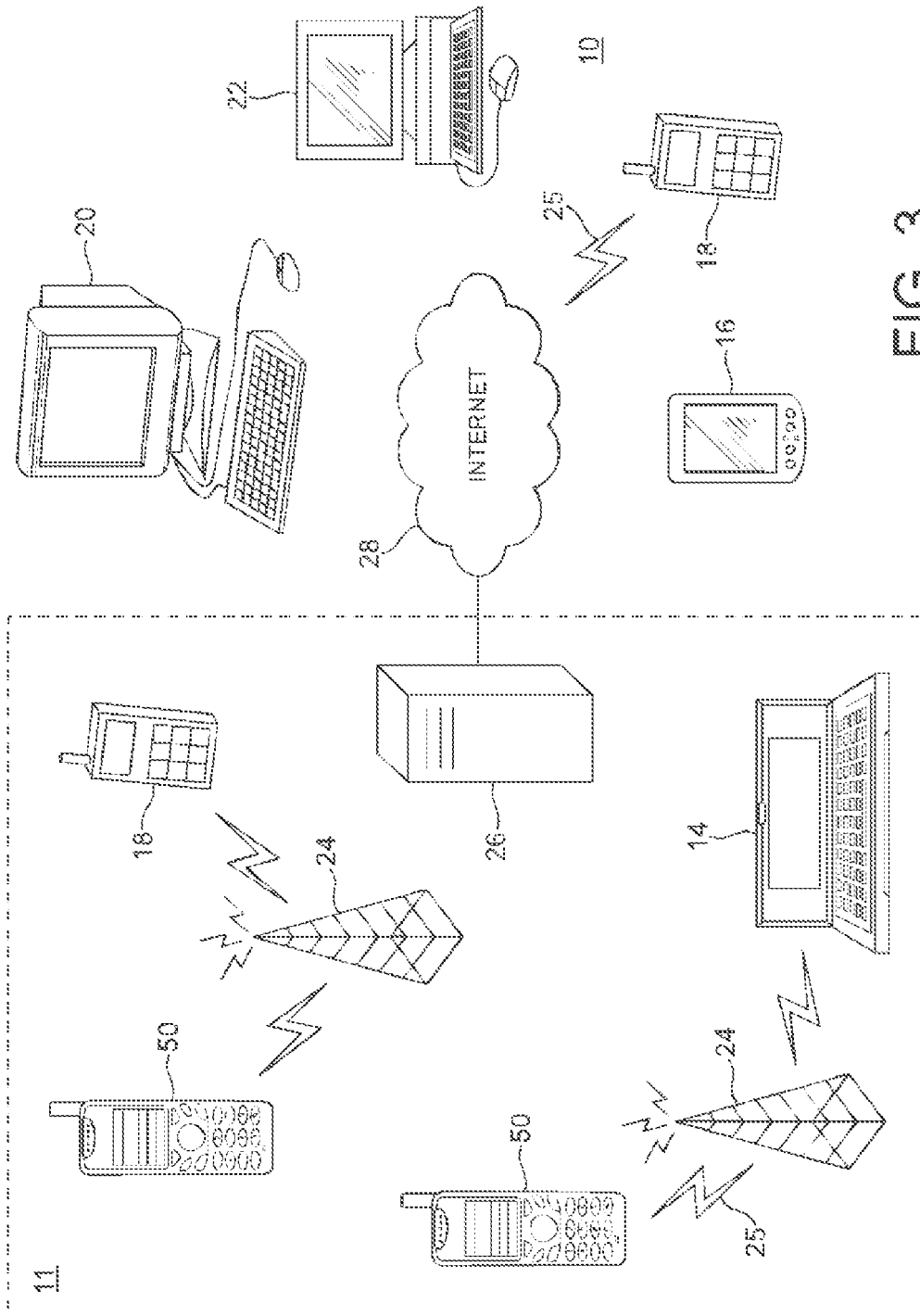


FIG. 2



3
5
L

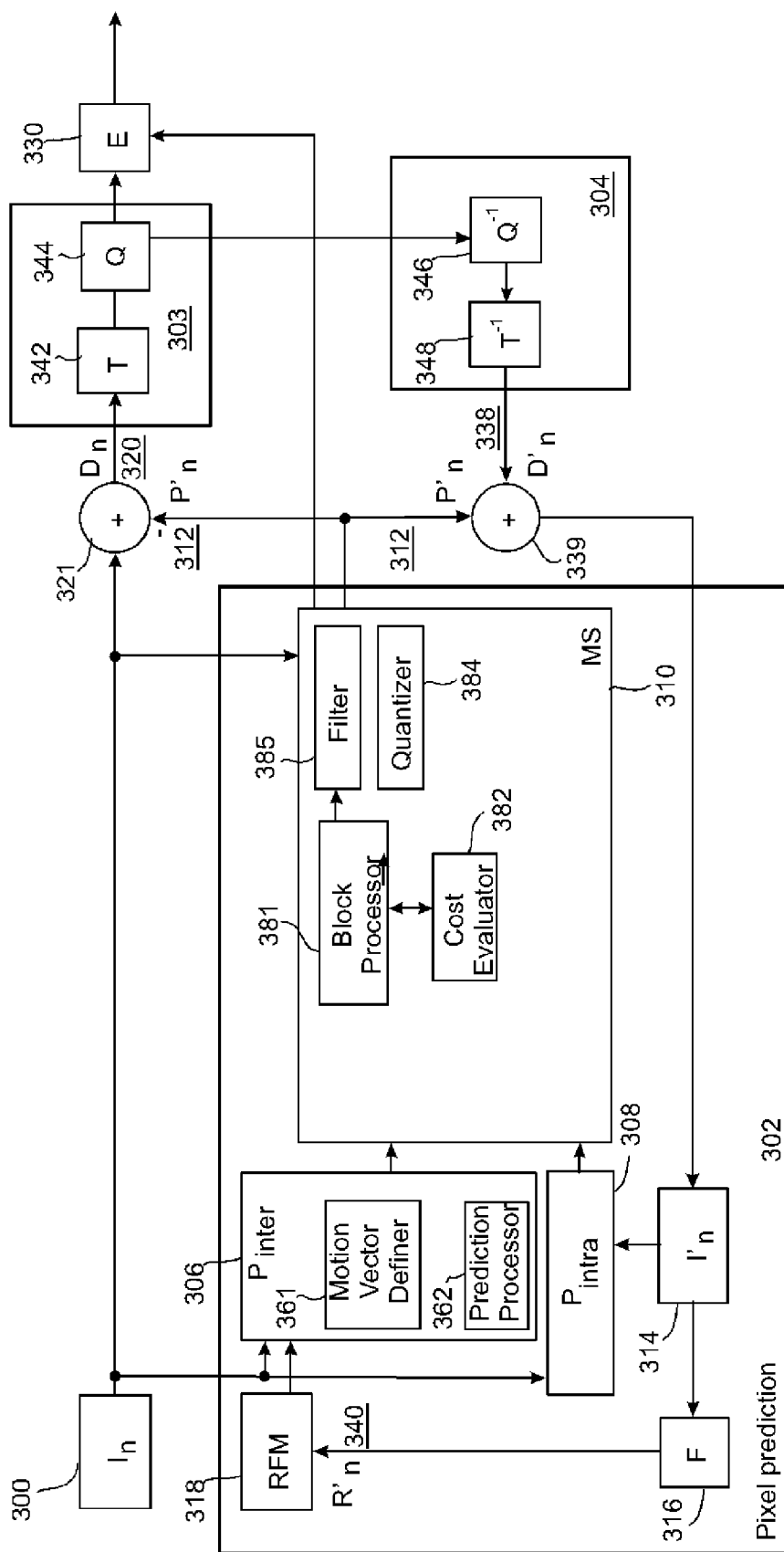


Fig. 4a

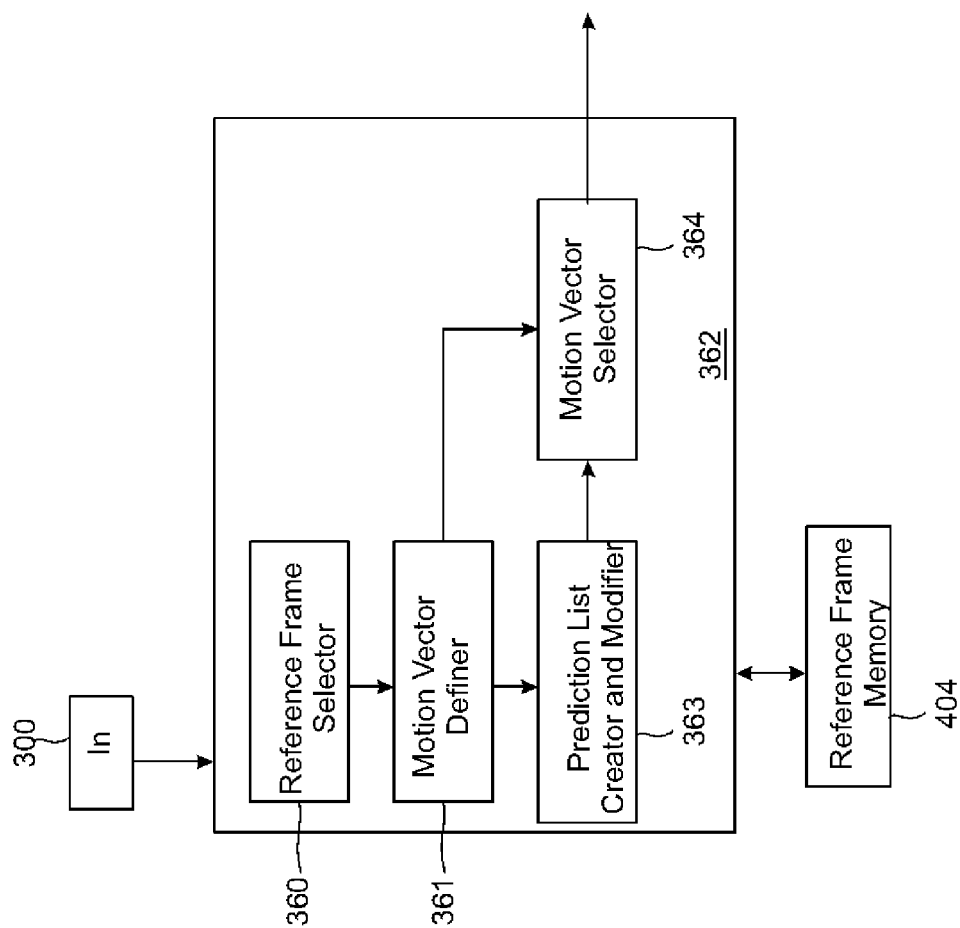


Fig. 4b

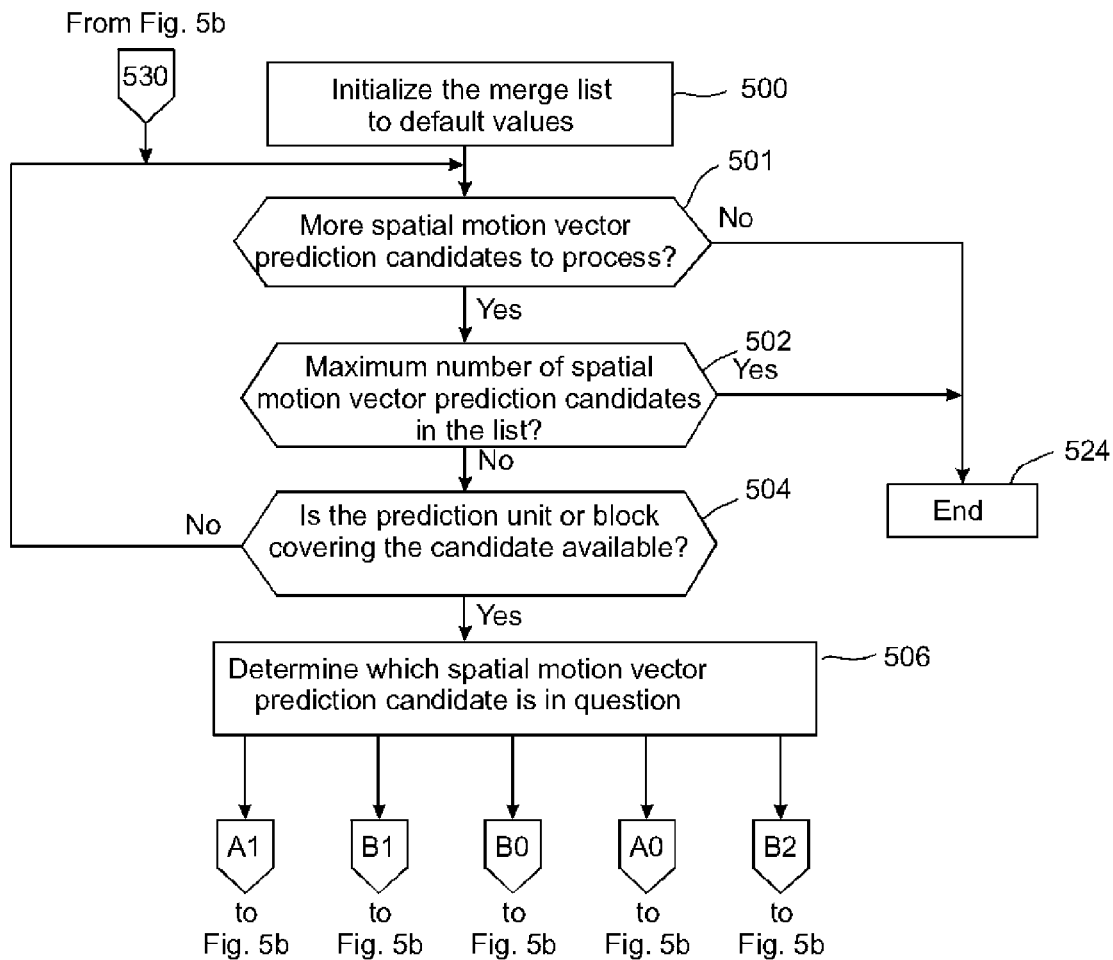


Fig. 5a

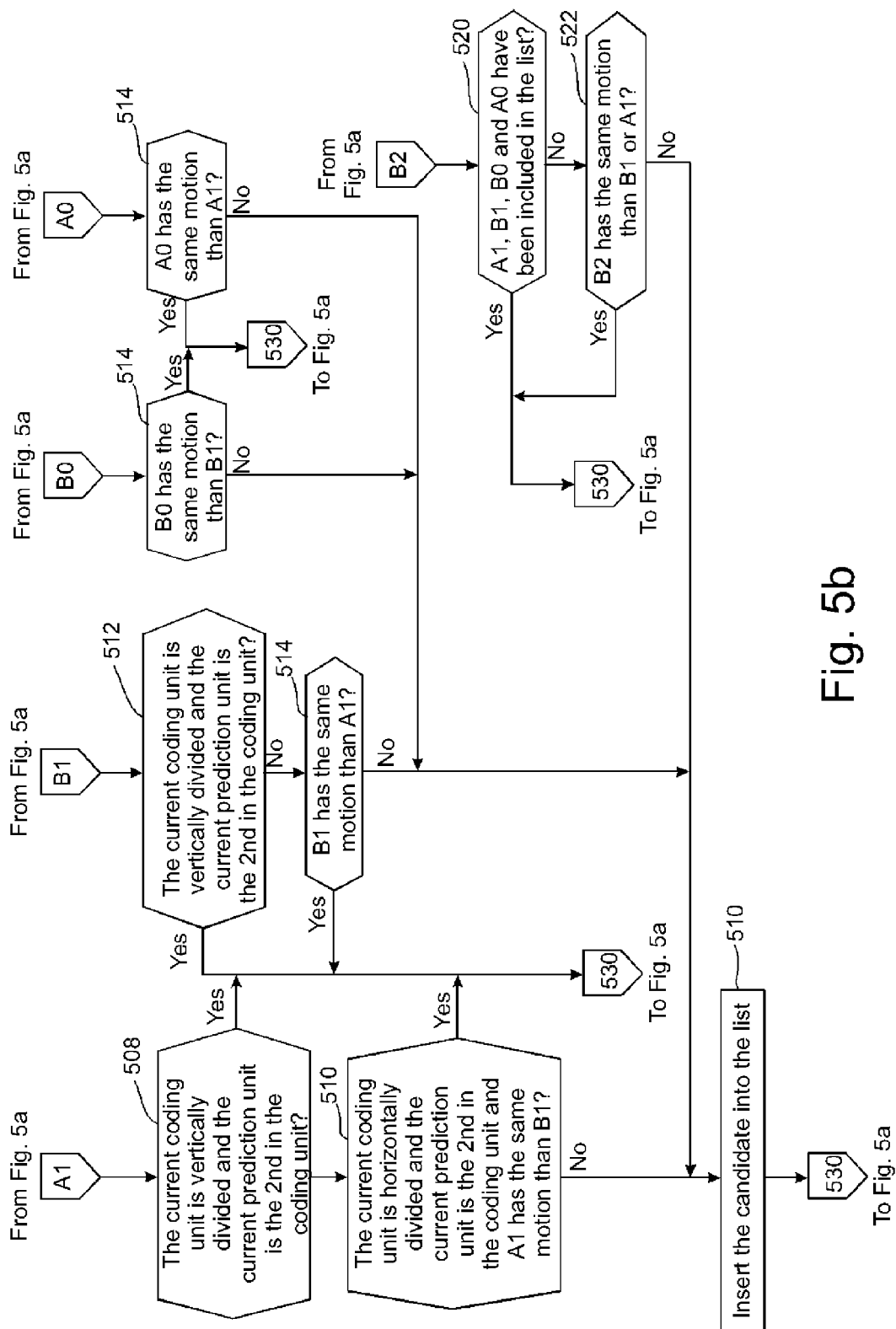


Fig. 5b

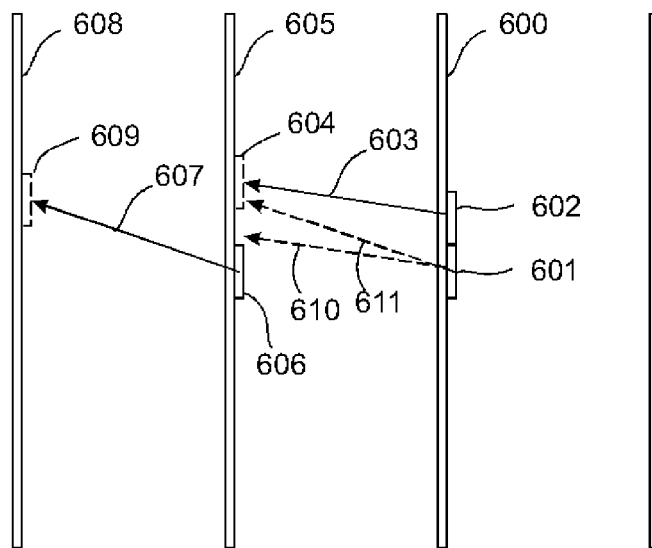


Fig. 6a

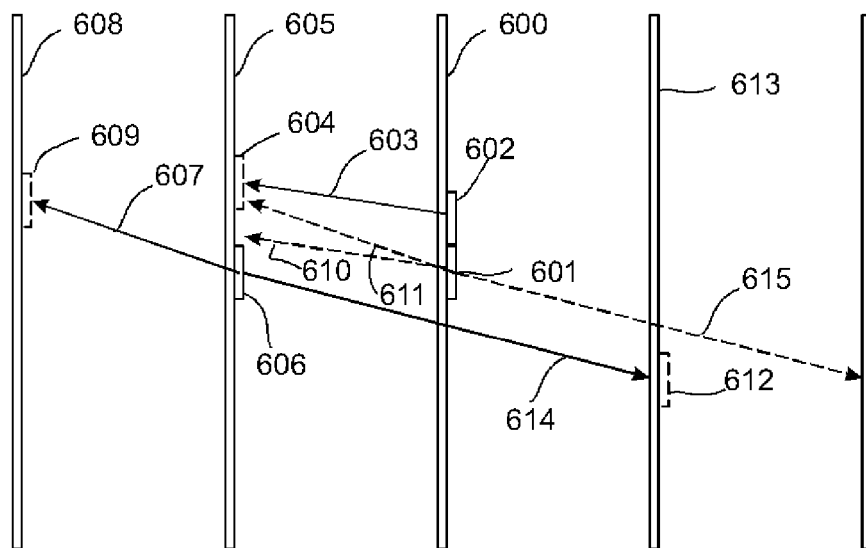


Fig. 6b

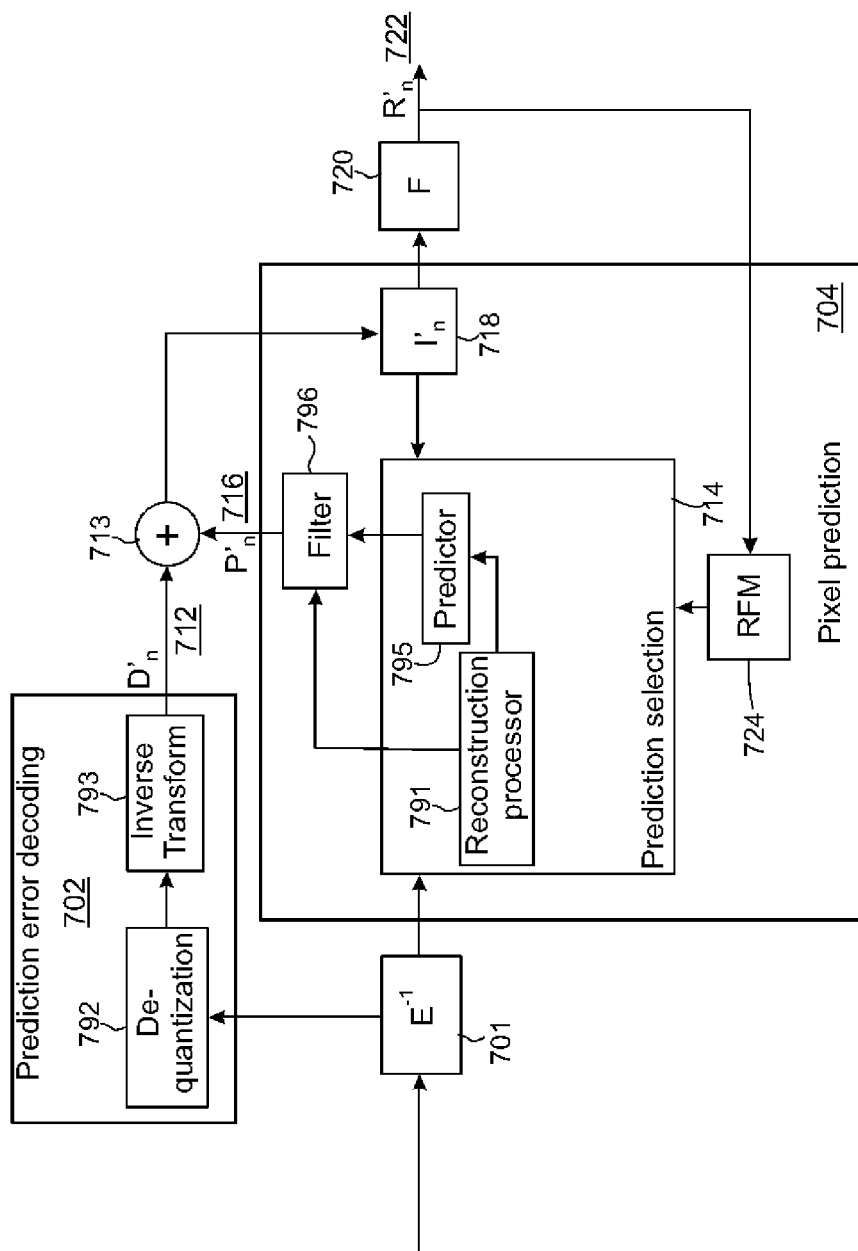


Fig. 7

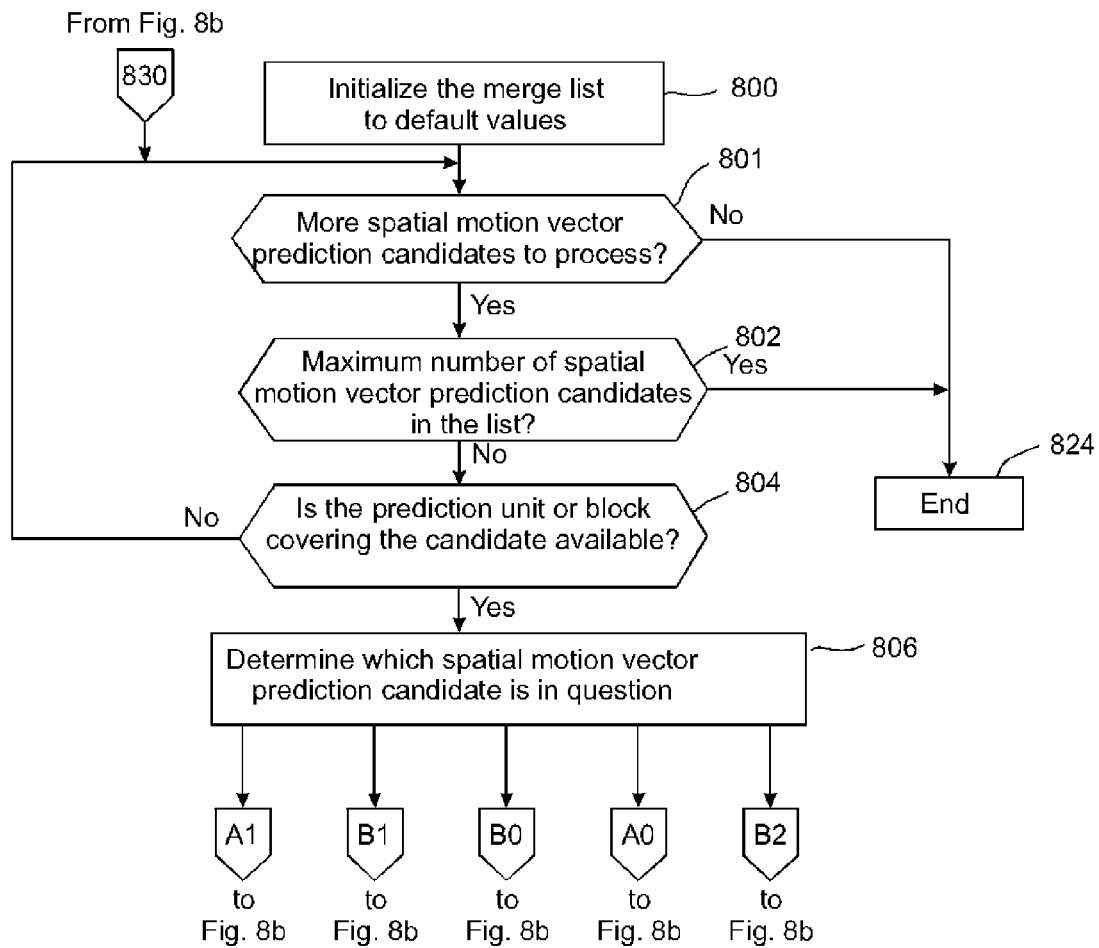


Fig. 8a

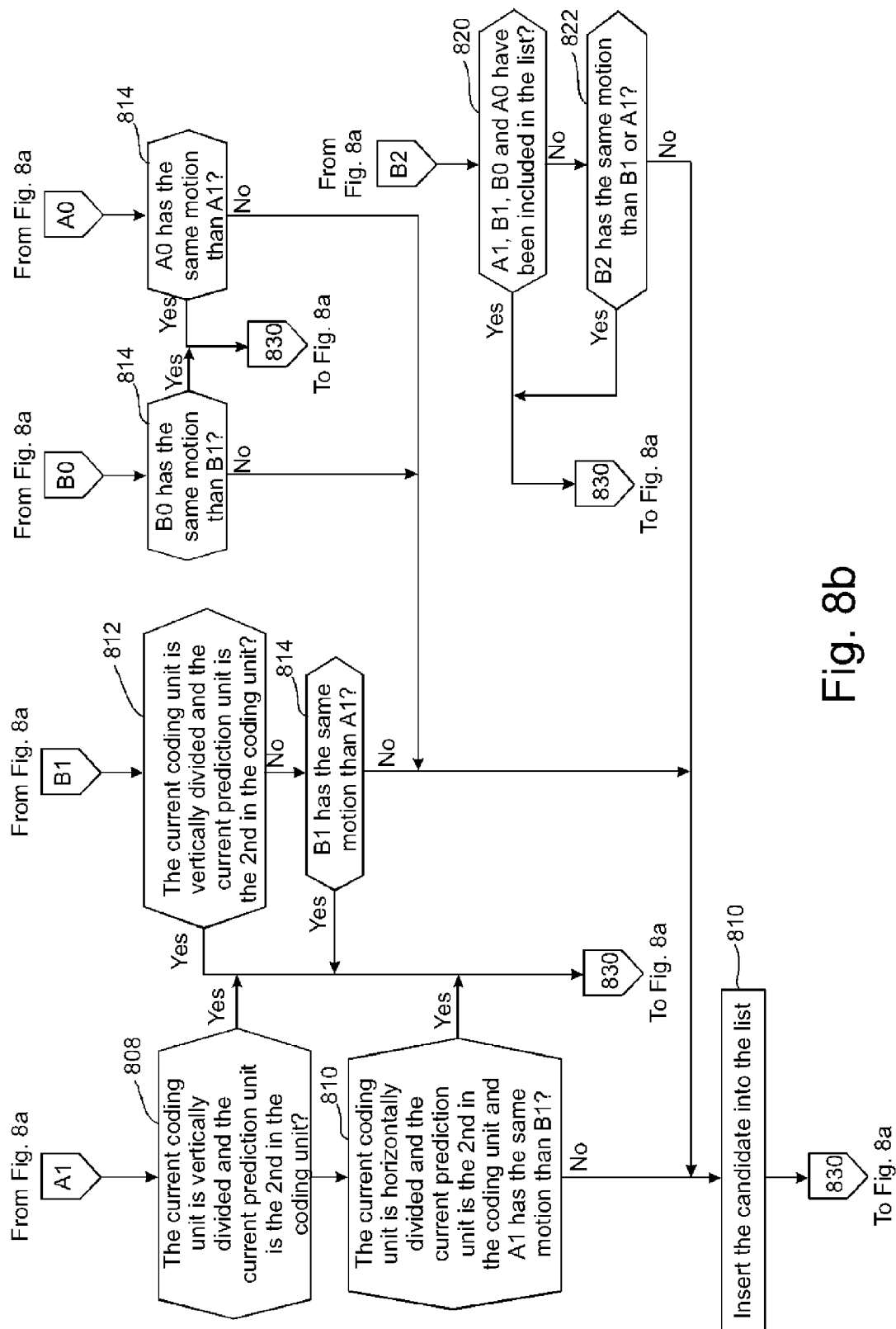


Fig. 8b

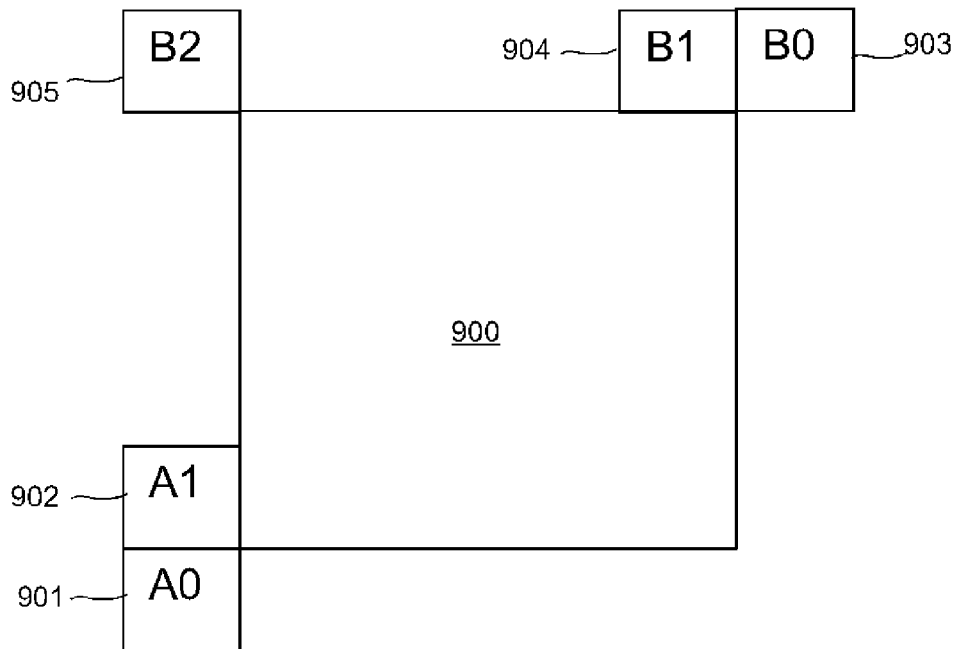


Fig. 9

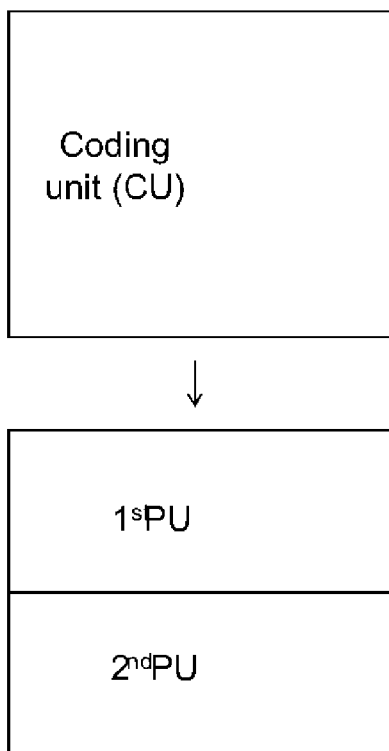


Fig. 10a

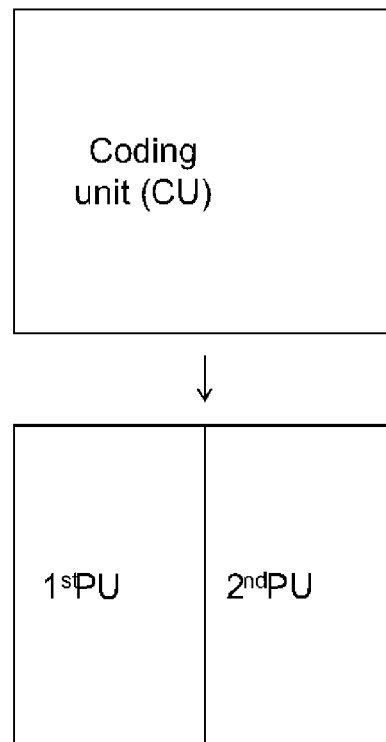


Fig. 10b

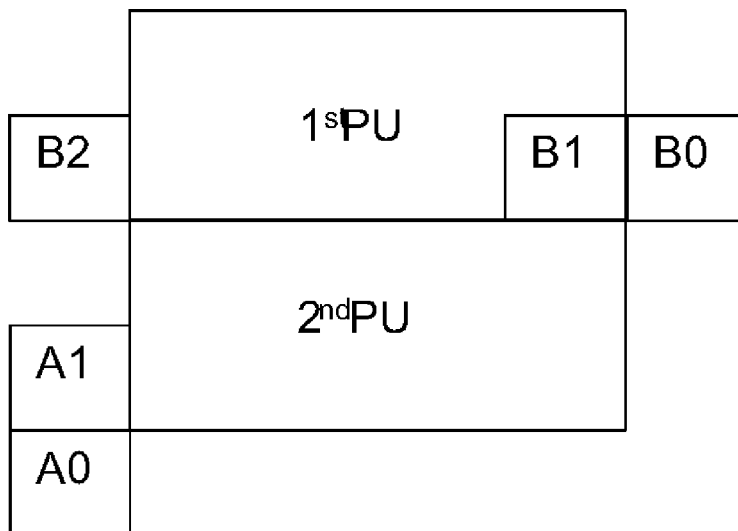


Fig. 11a

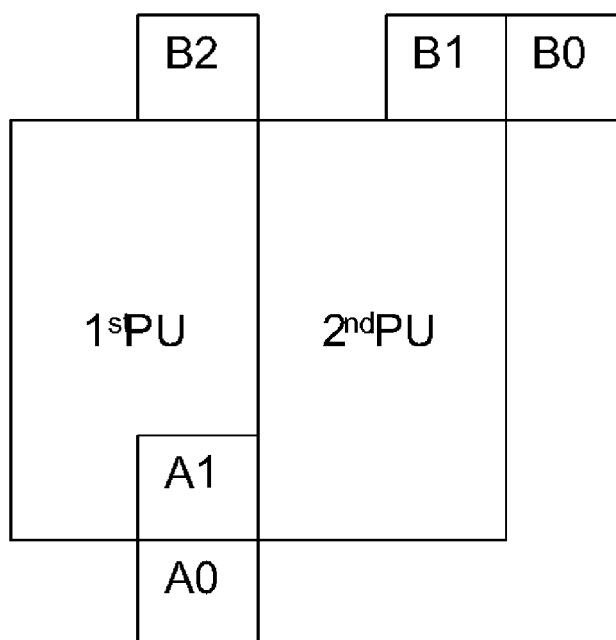


Fig. 11b

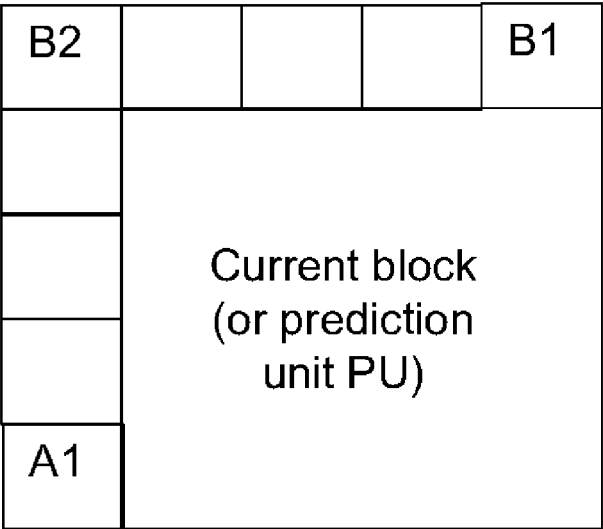


Fig. 12

METHOD FOR CODING AND AN APPARATUS

TECHNICAL FIELD

There is provided a method for encoding, a method for decoding, an apparatus, computer program products, an encoder and a decoder.

BACKGROUND INFORMATION

This section is intended to provide a background or context to the invention that is recited in the claims. The description herein may include concepts that could be pursued, but are not necessarily ones that have been previously conceived or pursued. Therefore, unless otherwise indicated herein, what is described in this section is not prior art to the description and claims in this application and is not admitted to be prior art by inclusion in this section

A video codec may comprise an encoder which transforms input video into a compressed representation suitable for storage and/or transmission and a decoder that can uncompress the compressed video representation back into a viewable form, or either one of them. The encoder may discard some information in the original video sequence in order to represent the video in a more compact form, for example at a lower bit rate.

Many hybrid video codecs, operating for example according to the International Telecommunication Union's ITU-T H.263 and H.264 coding standards, encode video information in two phases. In the first phase, pixel values in a certain picture area or "block" are predicted. These pixel values can be predicted, for example, by motion compensation mechanisms, which involve finding and indicating an area in one of the previously encoded video frames (or a later coded video frame) that corresponds closely to the block being coded. Additionally, pixel values can be predicted by spatial mechanisms which involve finding and indicating a spatial region relationship, for example by using pixel values around the block to be coded in a specified manner.

Prediction approaches using image information from a previous (or a later) image can also be called as Inter prediction methods, and prediction approaches using image information within the same image can also be called as Intra prediction methods.

The second phase is one of coding the error between the predicted block of pixels and the original block of pixels. This may be accomplished by transforming the difference in pixel values using a specified transform. This transform may be e.g. a Discrete Cosine Transform (DCT) or a variant thereof. After transforming the difference, the transformed difference may be quantized and entropy encoded.

By varying the fidelity of the quantization process, the encoder can control the balance between the accuracy of the pixel representation, (in other words, the quality of the picture) and the size of the resulting encoded video representation (in other words, the file size or transmission bit rate).

The decoder reconstructs the output video by applying a prediction mechanism similar to that used by the encoder in order to form a predicted representation of the pixel blocks (using the motion or spatial information created by the encoder and stored in the compressed representation of the image) and prediction error decoding (the inverse operation of the prediction error coding to recover the quantized prediction error signal in the spatial domain).

After applying pixel prediction and error decoding processes the decoder combines the prediction and the prediction error signals (the pixel values) to form the output video frame.

The decoder (and encoder) may also apply additional filtering processes in order to improve the quality of the output video before passing it for display and/or storing as a prediction reference for the forthcoming frames in the video sequence.

In some video codecs, such as High Efficiency Video Coding Working Draft 4, video pictures may be divided into coding units (CU) covering the area of a picture. A coding unit consists of one or more prediction units (PU) defining the prediction process for the samples within the coding unit and one or more transform units (TU) defining the prediction error coding process for the samples in the coding unit. A coding unit may consist of a square block of samples with a size selectable from a predefined set of possible coding unit sizes. A coding unit with the maximum allowed size can be named as a largest coding unit (LCU) and the video picture may be divided into non-overlapping largest coding units. A largest coding unit can further be split into a combination of smaller coding units, e.g. by recursively splitting the largest coding unit and resultant coding units. Each resulting coding unit may have at least one prediction unit and at least one transform unit associated with it. Each prediction unit and transform unit can further be split into smaller prediction units and transform units in order to increase granularity of the prediction and prediction error coding processes, respectively. Each prediction unit may have prediction information associated with it defining what kind of a prediction is to be applied for the pixels within that prediction unit (e.g. motion vector information for inter predicted prediction units and intra prediction directionality information for intra predicted prediction units). Similarly, each transform unit may be associated with information describing the prediction error decoding process for samples within the transform unit (including e.g. discrete cosine transform (DCT) coefficient information). It may be signalled at coding unit level whether prediction error coding is applied or not for each coding unit. In the case there is no prediction error residual associated with the coding unit, it can be considered there are no transform units for the coding unit. The division of the image into coding units, and division of coding units into prediction units and transform units may be signalled in the bitstream allowing the decoder to reproduce the intended structure of these units.

In some video codecs, motion information is indicated by motion vectors associated with each motion compensated image block. These motion vectors represent the displacement of the image block in the picture to be coded (in the encoder) or decoded (at the decoder) and the prediction source block in one of the previously coded or decoded images (or pictures). In order to represent motion vectors efficiently, motion vectors may be coded differentially with respect to block specific predicted motion vector. In some video codecs, the predicted motion vectors are created in a predefined way, for example by calculating the median of the encoded or decoded motion vectors of the adjacent blocks.

Another way to create motion vector predictions is to generate a list or a set of candidate predictions from blocks in the current frame and/or co-located or other blocks in temporal reference pictures and signalling the chosen candidate as the motion vector prediction. A spatial motion vector prediction is a prediction obtained only on the basis of information of one or more blocks of the same frame than

the current frame whereas temporal motion vector prediction is a prediction obtained on the basis of information of one or more blocks of a frame different from the current frame. It may also be possible to obtain motion vector predictions by combining both spatial and temporal prediction information of one or more encoded blocks. These kinds of motion vector predictions are called as spatio-temporal motion vector predictions.

In addition to predicting the motion vector values, the reference index in the reference picture list can be predicted. The reference index may be predicted from blocks in the current frame and/or co-located or other blocks in a temporal reference picture. Moreover, some high efficiency video codecs employ an additional motion information coding/decoding mechanism, often called merging/merge mode, where all the motion field information, which includes motion vector and corresponding reference picture index for each available reference picture list, may be predicted and used without any modification or correction. Similarly, predicting the motion field information may be carried out using the motion field information of blocks in the current frame and/or co-located or other blocks in temporal reference pictures and the used motion field information is signalled among a list of motion field candidate list filled with motion field information of available blocks in the current frame and/or co-located or other blocks in temporal reference pictures.

In some video codecs the prediction residual after motion compensation is first transformed with a transform kernel (like DCT) and then coded. The reason for this is that often there still exists some correlation among the residual and transform can in many cases help reduce this correlation and provide more efficient coding.

Some video encoders utilize Lagrangian cost functions to find optimal coding modes, e.g. the desired Macroblock mode and associated motion vectors. This kind of cost function uses a weighting factor λ to tie together the (exact or estimated) image distortion due to lossy coding methods and the (exact or estimated) amount of information that is required to represent the pixel values in an image area:

$$C=D+\lambda R \quad (1)$$

where C is the Lagrangian cost to be minimized, D is the image distortion (e.g. Mean Squared Error) with the mode and motion vectors considered, and R the number of bits needed to represent the required data to reconstruct the image block in the decoder (including the amount of data to represent the candidate motion vectors).

Some video codecs such as hybrid video codecs may generate a list of motion vector predictions (MVP) consisting of motion vectors of spatial adjacent blocks (spatial MVP) and/or motion vectors of blocks in a previously decoded frame (temporal MVP). One of the candidate motion vectors in the list is signalled to be used as the motion vector prediction of the current block. After the list is generated, some of the motion vector prediction candidates may have the same motion information. In this case, the identical motion vector prediction candidates may be removed to reduce redundancy. During the decoding, if the temporal motion vector prediction information is unavailable due to e.g. loss of reference frame, the decoder may not know if the temporal motion vector prediction candidate in the list is to be removed. This may lead to uncertainty for mapping the decoded candidate index to the candidates whose removal decision is based on comparing motion information with the temporal motion vector prediction. As a result, false assignment of motion vector prediction can-

didates may occur which may lead to degradation in the picture quality and drift of false motion information throughout the decoding process.

SUMMARY

The present invention introduces a method for generating a motion vector prediction list for an image block. In some embodiments video codecs employ in a motion prediction candidate list construction a way to reduce the complexity of the implementation. This can be achieved by performing a limited number of motion information comparisons between candidate pairs to remove the redundant candidates rather than comparing every available candidate pair. The decision of whether comparing two candidates may depend on the order of the candidates to be considered for the list and/or coding/prediction mode and/or location of the blocks associated with the candidates. In some embodiments a video codec employs a merge process for motion information coding and creates a list of motion prediction candidates from which one of the candidates is to be signalled as the motion information for the current coding or prediction unit. The motion prediction candidates may consist of several spatial motion predictions and a temporal motion prediction. The spatial candidates are obtained from the motion information of e.g. spatial neighbour blocks.

According to a first aspect of the present invention there is provided a method comprising:

receiving a block of pixels including a prediction unit; determining a set of spatial motion vector prediction candidates for the block of pixels; the spatial motion vector prediction candidates being provided with motion information;

selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit;

determining a subset of spatial motion vector predictions based on the location of the block associated with the first spatial motion vector prediction candidate;

comparing motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates;

if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list.

According to a second aspect of the present invention there is provided a method comprising:

receiving an encoded block of pixels including a prediction unit;

determining a set of spatial motion vector prediction candidates for the encoded block of pixels; the spatial motion vector prediction candidates being provided with motion information;

selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit;

determining a subset of spatial motion vector predictions based on the location of the block associated with the first spatial motion vector prediction candidate;

comparing motion information of the first spatial motion vector prediction candidate with motion information of

5

another spatial motion vector prediction candidate of the set of spatial motion vector prediction candidates;

if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list.

According to a third aspect of the present invention there is provided an apparatus comprising a processor and a memory including computer program code, the memory and the computer program code configured to, with the processor, cause the apparatus to:

receive a block of pixels

including a prediction unit;

determining a set of spatial motion vector prediction candidates for the block of pixels; the spatial motion vector prediction candidates being provided with motion information;

selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit;

determining a subset of spatial motion vector predictions based on the location of the block associated with the first spatial motion vector prediction candidate;

comparing motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates;

if at least one the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list.

According to a fourth aspect of the present invention there is provided an apparatus comprising a processor and a memory including computer program code, the memory and the computer program code configured to, with the processor, cause the apparatus to:

receive an encoded block of pixels

including a prediction unit;

determining a set of spatial motion vector prediction candidates for the encoded block of pixels; the spatial motion vector prediction candidates being provided with motion information;

selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit;

determining a subset of spatial motion vector predictions based on the location of the block associated with the first spatial motion vector prediction candidate;

comparing motion information of the first spatial motion vector prediction candidate with motion information of another spatial motion vector prediction candidate of the set of spatial motion vector prediction candidates;

if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list.

According to a fifth aspect of the present invention there is provided a storage medium having stored thereon a computer executable program code for use by an encoder, said program code comprises instructions for:

6

receiving a block of pixels including a prediction unit;

determining a set of spatial motion vector prediction candidates for the block of pixels; the spatial motion vector prediction candidates being provided with motion information;

select a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit;

determine a subset of spatial motion vector predictions based on the location of the block associated with the first spatial motion vector prediction candidate;

compare motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates;

exclude the first spatial motion vector prediction candidate from the merge list, if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other

According to a sixth aspect of the present invention there is provided a storage medium having stored thereon a computer executable program code for use by a decoder, said program code comprises instructions for:

receiving an encoded block of pixels including a prediction unit;

determining a set of spatial motion vector prediction candidates for the encoded block of pixels; the spatial motion vector prediction candidates being provided with motion information;

selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit;

determining a subset of spatial motion vector predictions based on the location of the block associated with the first spatial motion vector prediction candidate;

comparing motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates;

if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list.

According to a seventh aspect of the present invention there is provided an apparatus comprising:

means for receiving a block of pixels including a prediction unit;

means for determining a set of spatial motion vector prediction candidates for the block of pixels; the spatial motion vector prediction candidates being provided with motion information;

selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit;

determining a subset of spatial motion vector predictions based on the location of the block associated with the first spatial motion vector prediction candidate;

comparing motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates;

7

if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list.

According to an eighth aspect of the present invention there is provided an apparatus comprising:

means for receiving an encoded block of pixels including a prediction unit;

means for determining a set of spatial motion vector prediction candidates for the encoded block of pixels; the spatial motion vector prediction candidates being provided with motion information;

means for selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit;

means for determining a subset of spatial motion vector predictions based on the location of the block associated with the first spatial motion vector prediction candidate;

means for comparing motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates;

means for excluding the first spatial motion vector prediction candidate from the merge list, if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other.

DESCRIPTION OF THE DRAWINGS

For better understanding of the present invention, reference will now be made by way of example to the accompanying drawings in which:

FIG. 1 shows schematically an electronic device employing some embodiments of the invention;

FIG. 2 shows schematically a user equipment suitable for employing some embodiments of the invention;

FIG. 3 further shows schematically electronic devices employing embodiments of the invention connected using wireless and wired network connections;

FIG. 4a shows schematically an embodiment of the invention as incorporated within an encoder;

FIG. 4b shows schematically an embodiment of a prediction reference list generation and modification according to some embodiments of the invention;

FIGS. 5a and 5b show a flow diagram showing the operation of an embodiment of the invention with respect to the encoder as shown in FIG. 4a;

FIG. 6a illustrates an example of spatial and temporal prediction of a prediction unit;

FIG. 6b illustrates another example of spatial and temporal prediction of a prediction unit;

FIG. 7 shows schematically an embodiment of the invention as incorporated within a decoder;

FIGS. 8a and 8b show a flow diagram of showing the operation of an embodiment of the invention with respect to the decoder shown in FIG. 7;

FIG. 9 illustrates an example of a coding unit and some neighbour blocks of the coding unit;

FIG. 10a illustrates an example of a horizontal division of the coding unit;

FIG. 10b illustrates an example of a vertical division of the coding unit;

8

FIG. 11a illustrates locations of five spatial neighbours A0, A1, B0, B1, B2 for a prediction unit generated as the second prediction unit of a horizontally divided coding unit;

FIG. 11b illustrates locations of five spatial neighbours for a prediction unit generated as the second prediction unit of a vertically divided coding unit; and

FIG. 12 illustrates an example of blocks between some spatial neighbours of a coding unit.

DETAILED DESCRIPTION OF SOME EXAMPLE EMBODIMENTS

The following describes in further detail suitable apparatus and possible mechanisms for the provision of improving the prediction accuracy and hence possibly reducing information to be transmitted in video coding systems. In this regard reference is first made to FIG. 1 which shows a schematic block diagram of an exemplary apparatus or electronic device 50, which may incorporate a codec according to an embodiment of the invention.

The electronic device 50 may for example be a mobile terminal or user equipment of a wireless communication system. However, it would be appreciated that embodiments of the invention may be implemented within any electronic device or apparatus which may require encoding and decoding or encoding or decoding video images.

The apparatus 50 may comprise a housing 30 for incorporating and protecting the device. The apparatus 50 further may comprise a display 32 in the form of a liquid crystal display. In other embodiments of the invention the display may be any suitable display technology suitable to display an image or video. The apparatus 50 may further comprise a keypad 34. In other embodiments of the invention any suitable data or user interface mechanism may be employed. For example the user interface may be implemented as a virtual keyboard or data entry system as part of a touch-sensitive display. The apparatus may comprise a microphone 36 or any suitable audio input which may be a digital or analogue signal input. The apparatus 50 may further comprise an audio output device which in embodiments of the invention may be any one of: an earpiece 38, speaker, or an analogue audio or digital audio output connection. The apparatus 50 may also comprise a battery 40 (or in other embodiments of the invention the device may be powered by any suitable mobile energy device such as solar cell, fuel cell or clockwork generator). The apparatus may further comprise an infrared port 42 for short range line of sight communication to other devices. In other embodiments the apparatus 50 may further comprise any suitable short range communication solution such as for example a Bluetooth wireless connection or a USB/firewire wired connection.

The apparatus 50 may comprise a controller 56 or processor for controlling the apparatus 50. The controller 56 may be connected to memory 58 which in embodiments of the invention may store both data in the form of image and audio data and/or may also store instructions for implementation on the controller 56. The controller 56 may further be connected to codec circuitry 54 suitable for carrying out coding and decoding of audio and/or video data or assisting in coding and decoding carried out by the controller 56.

The apparatus 50 may further comprise a card reader 48 and a smart card 46, for example a UICC and UICC reader for providing user information and being suitable for providing authentication information for authentication and authorization of the user at a network.

The apparatus 50 may comprise radio interface circuitry 52 connected to the controller and suitable for generating

wireless communication signals for example for communication with a cellular communications network, a wireless communications system or a wireless local area network. The apparatus 50 may further comprise an antenna 44 connected to the radio interface circuitry 52 for transmitting radio frequency signals generated at the radio interface circuitry 52 to other apparatus(es) and for receiving radio frequency signals from other apparatus(es).

In some embodiments of the invention, the apparatus 50 comprises a camera capable of recording or detecting individual frames which are then passed to the codec 54 or controller for processing. In some embodiments of the invention, the apparatus may receive the video image data for processing from another device prior to transmission and/or storage. In some embodiments of the invention, the apparatus 50 may receive either wirelessly or by a wired connection the image for coding/decoding.

With respect to FIG. 3, an example of a system within which embodiments of the present invention can be utilized is shown. The system 10 comprises multiple communication devices which can communicate through one or more networks. The system 10 may comprise any combination of wired or wireless networks including, but not limited to a wireless cellular telephone network (such as a GSM, UMTS, CDMA network etc), a wireless local area network (WLAN) such as defined by any of the IEEE 802.x standards, a Bluetooth personal area network, an Ethernet local area network, a token ring local area network, a wide area network, and the Internet.

The system 10 may include both wired and wireless communication devices or apparatus 50 suitable for implementing embodiments of the invention.

For example, the system shown in FIG. 3 shows a mobile telephone network 11 and a representation of the internet 28. Connectivity to the internet 28 may include, but is not limited to, long range wireless connections, short range wireless connections, and various wired connections including, but not limited to, telephone lines, cable lines, power lines, and similar communication pathways.

The example communication devices shown in the system 10 may include, but are not limited to, an electronic device or apparatus 50, a combination of a personal digital assistant (PDA) and a mobile telephone 14, a PDA 16, an integrated messaging device (IMD) 18, a desktop computer 20, a notebook computer 22. The apparatus 50 may be stationary or mobile when carried by an individual who is moving. The apparatus 50 may also be located in a mode of transport including, but not limited to, a car, a truck, a taxi, a bus, a train, a boat, an airplane, a bicycle, a motorcycle or any similar suitable mode of transport.

Some or further apparatuses may send and receive calls and messages and communicate with service providers through a wireless connection 25 to a base station 24. The base station 24 may be connected to a network server 26 that allows communication between the mobile telephone network 11 and the internet 28. The system may include additional communication devices and communication devices of various types.

The communication devices may communicate using various transmission technologies including, but not limited to, code division multiple access (CDMA), global systems for mobile communications (GSM), universal mobile telecommunications system (UMTS), time divisional multiple access (TDMA), frequency division multiple access (FDMA), transmission control protocol-internet protocol (TCP-IP), short messaging service (SMS), multimedia messaging service (MMS), email, instant messaging service

(IMS), Bluetooth, IEEE 802.11 and any similar wireless communication technology. A communications device involved in implementing various embodiments of the present invention may communicate using various media including, but not limited to, radio, infrared, laser, cable connections, and any suitable connection.

With respect to FIG. 4a, a block diagram of a video encoder suitable for carrying out embodiments of the invention is shown. Furthermore, with respect to FIGS. 5a and 5b, the operation of the encoder exemplifying embodiments of the invention specifically with respect to construction of the list of candidate predictions is shown as a flow diagram.

FIG. 4a shows the encoder as comprising a pixel predictor 302, prediction error encoder 303 and prediction error decoder 304. FIG. 4a also shows an embodiment of the pixel predictor 302 as comprising an inter-predictor 306, an intra-predictor 308, a mode selector 310, a filter 316, and a reference frame memory 318. In this embodiment the mode selector 310 comprises a block processor 381 and a cost evaluator 382. The encoder may further comprise an entropy encoder 330 for entropy encoding the bit stream.

FIG. 4b depicts an embodiment of the inter predictor 306. The inter predictor 306 comprises a reference frame selector 360 for selecting reference frame or frames, a motion vector definer 361, a prediction list modifier 363 and a motion vector selector 364. These elements or some of them may be part of a prediction processor 362 or they may be implemented by using other means.

The pixel predictor 302 receives the image 300 to be encoded at both the inter-predictor 306 (which determines the difference between the image and a motion compensated reference frame 318) and the intra-predictor 308 (which determines a prediction for an image block based only on the already processed parts of the current frame or picture). The output of both the inter-predictor and the intra-predictor may be passed to the mode selector 310. The intra-predictor 308 may have more than one intra-prediction modes. Hence, each mode may perform the intra-prediction and provide the predicted signal to the mode selector 310. The mode selector 310 also receives a copy of the image 300.

The mode selector 310 determines which encoding mode to use to encode the current block. If the mode selector 310 decides to use an inter-prediction mode it will pass the output of the inter-predictor 306 to the output of the mode selector 310. If the mode selector 310 decides to use an intra-prediction mode it will pass the output of one of the intra-predictor modes to the output of the mode selector 310.

The output of the mode selector is passed to a first summing device 321. The first summing device may subtract the pixel predictor 302 output from the image 300 to produce a first prediction error signal 320 which is input to the prediction error encoder 303.

The pixel predictor 302 further receives from a preliminary reconstructor 339 the combination of the prediction representation of the image block 312 and the output 338 of the prediction error decoder 304. The preliminary reconstructed image 314 may be passed to the intra-predictor 308 and to a filter 316. The filter 316 receiving the preliminary representation may filter the preliminary representation and output a final reconstructed image 340 which may be saved in a reference frame memory 318. The reference frame memory 318 may be connected to the inter-predictor 306 to be used as the reference image against which the future image 300 is compared in inter-prediction operations.

The operation of the pixel predictor 302 may be configured to carry out any known pixel prediction algorithm known in the art.

The pixel predictor **302** may also comprise a filter **385** to filter the predicted values before outputting them from the pixel predictor **302**.

The operation of the prediction error encoder **302** and prediction error decoder **304** will be described hereafter in further detail. In the following examples the encoder generates images in terms of 16×16 pixel macroblocks which go to form the full image or picture. Thus, for the following examples the pixel predictor **302** outputs a series of predicted macroblocks of size 16×16 pixels and the first summing device **321** outputs a series of 16×16 pixel residual data macroblocks which may represent the difference between a first macro-block in the image **300** against a predicted macro-block (output of pixel predictor **302**). It would be appreciated that other size macro blocks may be used.

The prediction error encoder **303** comprises a transform block **342** and a quantizer **344**. The transform block **342** transforms the first prediction error signal **320** to a transform domain. The transform is, for example, the DCT transform. The quantizer **344** quantizes the transform domain signal, e.g. the DCT coefficients, to form quantized coefficients.

The prediction error decoder **304** receives the output from the prediction error encoder **303** and performs the opposite processes of the prediction error encoder **303** to produce a decoded prediction error signal **338** which when combined with the prediction representation of the image block **312** at the second summing device **339** produces the preliminary reconstructed image **314**. The prediction error decoder may be considered to comprise a dequantizer **346**, which dequantizes the quantized coefficient values, e.g. DCT coefficients, to reconstruct the transform signal and an inverse transformation block **348**, which performs the inverse transformation to the reconstructed transform signal wherein the output of the inverse transformation block **348** contains reconstructed block(s). The prediction error decoder may also comprise a macroblock filter (not shown) which may filter the reconstructed macroblock according to further decoded information and filter parameters.

In the following the operation of an example embodiment of the inter predictor **306** will be described in more detail. The inter predictor **306** receives the current block for inter prediction. It is assumed that for the current block there already exists one or more neighbouring blocks which have been encoded and motion vectors have been defined for them. For example, the block on the left side and/or the block above the current block may be such blocks. Spatial motion vector predictions for the current block can be formed e.g. by using the motion vectors of the encoded neighbouring blocks and/or of non-neighbour blocks in the same slice or frame, using linear or non-linear functions of spatial motion vector predictions, using a combination of various spatial motion vector predictors with linear or non-linear operations, or by any other appropriate means that do not make use of temporal reference information. It may also be possible to obtain motion vector predictors by combining both spatial and temporal prediction information of one or more encoded blocks. These kinds of motion vector predictors may also be called as spatio-temporal motion vector predictors.

Reference frames used in encoding the neighbouring blocks have been stored to the reference frame memory **404**. The reference frames may be short term references or long term references and each reference frame may have a unique index indicative of the location of the reference frame in the reference frame memory. When a reference frame is no longer used as a reference frame it may be removed from the

reference frame memory or marked as a non-reference frame wherein the storage location of that reference frame may be occupied for a new reference frame. In addition to the reference frames of the neighbouring blocks the reference frame selector **360** may also select one or more other frames as potential reference frames and store them to the reference frame memory.

Motion vector information of encoded blocks is also stored into the memory so that the inter predictor **306** is able to retrieve the motion vector information when processing motion vector candidates for the current block.

In some embodiments the motion vectors are stored into one or more lists. For example, motion vectors of unidirectionally predicted frames (e.g. P-frames) may be stored to a list called as list 0. For bi-directionally predicted frames (e.g. B-frames) there may be two lists (list 0 and list 1) and for multi-predicted frames there may be more than two lists. Reference frame indices possibly associated with the motion vectors may also be stored in one or more lists.

In some embodiments there may be two or more motion vector prediction procedures and each procedure may have its own candidate set creation process. In one procedure, only the motion vector values are used. In another procedure, which may be called as a Merge Mode, each candidate element may comprise 1) The information whether 'block was uni-predicted using only list0' or 'block was uni-predicted using only list1' or 'block was bi-predicted using list0 and list1' 2) motion vector value for list0 3) Reference picture index in list0 4) motion vector value for list1 5) Reference picture index list1. Therefore, whenever two prediction candidates are to be compared, not only the motion vector values are compared, but also the five values mentioned above may be compared to determine whether they correspond with each other or not. On the other hand, if any of the comparisons indicate that the prediction candidates do not have equal motion information, no further comparisons need be performed.

The motion vector definer **361** defines candidate motion vectors for the current frame by using one or more of the motion vectors of one or more neighbour blocks and/or other blocks of the current block in the same frame and/or co-located blocks and/or other blocks of the current block in one or more other frames. These candidate motion vectors can be called as a set of candidate predictors or a predictor set. Each candidate predictor thus represents the motion vector of one or more already encoded block. In some embodiments the motion vector of the candidate predictor is set equal to the motion vector of a neighbour block for the same list if the current block and the neighbour block refer to the same reference frames for that list. Also for temporal prediction there may be one or more previously encoded frames wherein motion vectors of a co-located block or other blocks in a previously encoded frame can be selected as candidate predictors for the current block. The temporal motion vector predictor candidate can be generated by any means that make use of the frames other than the current frame.

The candidate motion vectors can also be obtained by using more than one motion vector of one or more other blocks such as neighbour blocks of the current block and/or co-located blocks in one or more other frames. As an example, any combination of the motion vector of the block to the left of the current block, the motion vector of the block above the current block, and the motion vector of the block at the up-right corner of the current block may be used (i.e. the block to the right of the block above the current block). The combination may be a median of the motion vectors or

13

calculated by using other formulas. For example, one or more of the motion vectors to be used in the combination may be scaled by a scaling factor, an offset may be added, and/or a constant motion vector may be added. In some embodiments the combined motion vector is based on both temporal and spatial motion vectors, e.g. the motion vector of one or more of the neighbour block or other block of the current block and the motion vector of a co-located block or other block in another frame.

If a neighbour block does not have any motion vector information a default motion vector such as a zero motion vector may be used instead.

FIG. 9 illustrates an example of a coding unit 900 and some neighbour blocks 901-905 of the coding unit. As can be seen from FIG. 9, if the coding unit 900 represents the current block, the neighbouring blocks 901-905 labelled A0, A1, B0, B1 and B2 could be such neighbour blocks which may be used when obtaining the candidate motion vectors.

Creating additional or extra motion vector predictions based on previously added predictors may be needed when the current number of candidates is limited or insufficient. This kind of creating additional candidates can be performed by combining previous two predictions and/or processing one previous candidate by scaling or adding offset and/or adding a zero motion vector with various reference indices. Hence, the motion vector definer 361 may examine how many motion vector candidates can be defined and how many potential candidate motion vectors exist for the current block. If the number of potential motion vector candidates is smaller than a threshold, the motion vector definer 361 may create additional motion vector predictions.

In some embodiments the combined motion vector can be based on motion vectors in different lists. For example, one motion vector may be defined by combining one motion vector from the list 0 and one motion vector from the list 1 e.g. when the neighbouring or co-located block is a bi-directionally predicted block and there exists one motion vector in the list 0 and one motion vector in the list 1 for the bi-directionally predicted block.

To distinguish the current block from the encoded/decoded blocks the motion vectors of which are used as candidate motion vectors, those encoded/decoded blocks are also called as reference blocks in this application.

In some embodiments not only the motion vector information of the reference block(s) is obtained (e.g. by copying) but also a reference index of the reference block in the reference picture list may be copied to the candidate list. The information whether the block was uni-predicted using only list0 or the block was uni-predicted using only list1 or the block was bi-predicted using list0 and list1 may also be copied. The candidate list may also be called as a candidate set or a set of motion vector prediction candidates.

FIG. 6a illustrates an example of spatial and temporal prediction of a prediction unit. There is depicted the current block 601 in the frame 600 and a neighbour block 602 which already has been encoded. The motion vector definer 361 has defined a motion vector 603 for the neighbour block 602 which points to a block 604 in the previous frame 605. This motion vector can be used as a potential spatial motion vector prediction 610 for the current block. FIG. 6a depicts that a co-located block 606 in the previous frame 605, i.e. the block at the same location than the current block but in the previous frame, has a motion vector 607 pointing to a block 609 in another frame 608. This motion vector 607 can be used as a potential temporal motion vector prediction-611 for the current frame.

14

FIG. 6b illustrates another example of spatial and temporal prediction of a prediction unit. In this example the block 606 of the previous frame 605 uses bi-directional prediction based on the block 609 of the frame preceding the frame 605 and on the block 612 succeeding the current frame 600. The temporal motion vector prediction for the current block 601 may be formed by using both the motion vectors 607, 614 or either of them.

The operation of the prediction list modifier 363 will now be described in more detail with reference to the flow diagram of FIGS. 5a and 5b. The prediction list modifier 363 initializes a motion vector prediction list to default values in block 500 of FIG. 5a. The prediction list modifier 363 may also initialize a list index to an initial value such as zero. Then, in block 501 the prediction list modifier checks whether there are any motion vector candidates to process. If there is at least one motion vector candidate in the predictor set for processing, the prediction list modifier 363 generates the next motion vector candidate which may be a temporal motion vector or a spatial motion vector. The comparison can be an identity/equivalence check or comparing the (absolute) difference against a threshold or any other similarity metric.

In the following, a merge process for motion information coding according to an example embodiment will be described in more detail. The encoder creates a list of motion prediction candidates from which one of the candidates is to be signalled as the motion information for the current coding unit or prediction unit. The motion prediction candidates may consist of several spatial motion predictions and a temporal motion prediction. The spatial candidates can be obtained from the motion information of e.g. the spatial neighbour blocks A0, A1, B0, B1, B2, whose motion information is used as spatial candidate motion predictions. The temporal motion prediction candidate may be obtained by processing the motion of a block in a frame other than the current frame. In this example embodiment, the encoder operations to construct the merge list for the spatial candidates may include the following. The operations may be carried out by the prediction list modifier 363, for example.

A maximum number of spatial motion prediction candidates to be included in the merge list may be defined. This maximum number may have been stored, for example, to the memory 58 of the apparatus 50, or to another appropriate place. It is also possible to determine the maximum number by using other means, or it may be determined in the software of the encoder of the apparatus 50.

In some embodiments the maximum number of spatial motion prediction candidates to be included in the merge list is four but in some embodiments the maximum number may be less than four or greater than four.

In this example the spatial motion prediction candidates are the spatial neighbour blocks A0, A1, B0, B1, B2. The spatial motion vector prediction candidate A1 is located on the left side of the prediction unit when the encoding/decoding order is from left to right and from top to bottom of the frame, slice or another entity to be encoded/decoded. Respectively, the spatial motion vector prediction candidate B1 is located above the prediction unit. third; the spatial motion vector prediction candidate B0 is on the right side of the spatial motion vector prediction candidate B1; the spatial motion vector prediction candidate A0 is below the spatial motion vector prediction candidate A1; and the spatial motion vector prediction candidate B2 is located on the same column than spatial motion vector prediction candidate A1 and on the same row than the spatial motion vector prediction candidate B1. In other words, the spatial motion

15

vector prediction candidate B2 is cornerwise neighbouring the prediction unit as can be seen e.g. from FIG. 9.

These spatial motion prediction candidates can be processed in a predetermined order, for example, A1, B1, B0, A0 and B2. The first spatial motion prediction candidate to be selected for further examination is thus A1. Before further examination is performed for the selected spatial motion prediction candidate, it may be determined whether the merge list already contains a maximum number of spatial motion prediction candidates. Hence, the prediction list modifier 363 compares 502 the number of spatial motion prediction candidates in the merge list with the maximum number, and if the number of spatial motion prediction candidates in the merge list is not less than the maximum number, the selected spatial motion prediction candidate is not included in the merge list and the process of constructing the merge list can be stopped 526. On the other hand, if the number of spatial motion prediction candidates in the merge list is less than the maximum number, a further analyses of the selected spatial motion prediction candidate is performed (blocks 504-522).

For all the spatial motion prediction candidates for which the further analyses is to be performed, some or all of the following conditions below may be tested for determining whether to include the spatial motion prediction candidate in the merge list.

The prediction list modifier 363 examines 504 if the prediction unit or block covering the spatial motion prediction candidate block is not available for motion prediction. If so, the candidate is not included in the merge list. The reason that the block is not available may be that the block is either coded in intra mode or resides in a different slice or outside of the picture area.

In addition to the common conditions above, for each spatial motion prediction candidate, if any of the following conditions holds, then the candidate is not included in the merge list, otherwise, it is included.

The prediction list modifier 363 determines 506 which spatial motion prediction candidate of the set of spatial motion prediction candidates is in question. If the spatial motion prediction candidate is the block A1, one or more of the following conditions may be examined 508, 510 to determine whether to include this spatial motion prediction candidate in the merge list or not. If the current coding unit 100 is vertically split into two rectangle prediction units 103, 104 as depicted in FIG. 10b and the current prediction unit is the second prediction unit 104 in the coding/decoding order (508), this spatial motion prediction candidate is not included in the merge list. If the current coding unit 100 is not vertically split into two rectangle prediction units but it is horizontally split into two rectangle prediction units 101, 102 as depicted in FIG. 10a and the current prediction unit is the second prediction unit in the coding/decoding order and the block A1 has the same motion information as the block B1 (510), this spatial motion prediction candidate (block A1) is not included in the merge list. In the example of FIG. 10a the second prediction unit is the lower prediction unit 102 of the coding unit 100 and in the example of FIG. 10b the second prediction unit is the rightmost prediction unit 104 of the coding unit 100. If none of the conditions above is fulfilled the block A1 is included in the merge list as a spatial motion prediction candidate (524).

If the spatial motion prediction candidate is the block B1, one or more of the following conditions may be examined 512, 514 to determine whether to include this spatial motion prediction candidate in the merge list or not. If the current coding unit 100 is horizontally split into two rectangle

16

prediction units 101, 102 as depicted in FIG. 10a and the current prediction unit is the second prediction unit 104 in the coding/decoding order (512), this spatial motion prediction candidate is not included in the merge list. If the current coding unit 100 is not horizontally split into two rectangle prediction units and if the block B1 has the same motion information than the block A1 (514), this spatial motion prediction candidate (block B1) is not included in the merge list. If none of the conditions above is fulfilled the block B1 is included in the merge list as a spatial motion prediction candidate (524).

If the spatial motion prediction candidate is the block B0, this spatial motion prediction candidate is not included in the merge list if the block B0 has the same motion information than the block B1 (516). Otherwise, if the number of spatial motion prediction candidates in the merge list is less than the maximum number of spatial motion prediction candidates, this spatial motion prediction candidate (block B0) is included in the merge list (524).

If the spatial motion prediction candidate is the block A0, this spatial motion prediction candidate is not included in the merge list if the block A0 has the same motion information than the block A1 (518). Otherwise, if the number of spatial motion prediction candidates in the merge list is less than the maximum number of spatial motion prediction candidates, this spatial motion prediction candidate (block A0) is included in the merge list (524).

If the spatial motion prediction candidate is the block B2, this spatial motion prediction candidate is not included in the merge list if the maximum number of spatial motion prediction candidates is four and the other blocks A0, A1, B0, and B1 are all decided to be included in the merge list (520). Otherwise, if the number of spatial motion prediction candidates in the merge list is less than the maximum number of spatial motion prediction candidates, the block B2 is not included in the merge list if the block B2 has the same motion information than the block B1 or the block A1 (522).

Then, after processing the blocks A1, B1, B0, A0 and B2 and including a subset of them in the merge list based on the above described conditions, no more redundancy check between these candidates are performed and remaining temporal motion prediction candidate and/or other possible additional candidates may be processed.

Comparing two blocks whether they have the same motion may be performed by comparing all the elements of the motion information, namely 1) The information whether 'the prediction unit is uni-predicted using only reference picture list0' or 'the prediction unit is un-predicted using only reference picture list1' or 'the prediction unit is bi-predicted using both reference picture list0 and list1' 2) Motion vector value corresponding to the reference picture list0 3) Reference picture index in the reference picture list0 4) Motion vector value corresponding to the reference picture list1 5) Reference picture index in the reference picture list1.

In some embodiments similar restrictions for comparing candidate pairs can be applied if the current coding unit is coded/decoded by splitting into four or any number of prediction units.

The maximum number of merge list candidates can be any non-zero value. In the example above the merger list candidates were the spatial neighbour blocks A0, A1, B0, B1, B2 and the temporal motion prediction candidate, but there may be more than one temporal motion prediction candidate and also other spatial motion prediction candidates than the

spatial neighbour blocks. In some embodiments there may also be other spatial neighbour blocks than the blocks A0, A1, B0, B1, B2.

It is also possible that the maximum number of spatial motion prediction candidates included in the list can be different than four.

In some embodiments the maximum number of merge list candidates and maximum number of spatial motion prediction candidates included in the list can depend on whether a temporal motion vector candidate is included in the list or not.

A different number of spatial motion prediction candidates located at various locations in the current frame can be processed. The locations can be the same as or different than A1, B1, B0, A0 and B2.

The decision of including which spatial motion prediction candidates in the list can be realized in two steps. In the first step, some of the candidates are eliminated by checking whether the candidate block is available and/or the candidate block's prediction mode is intra and/or whether the current block is a second prediction unit of a coding unit coded with two prediction units and the candidate has the same motion with the first prediction unit. In the second step, remaining candidates are examined and some or all of them are included in the merge list. The examination in the second step does not include comparing motion information of each possible candidate pair but includes a subset of the possible comparison combinations.

The decisions for the candidates can be taken in any order of A1, B1, B0, A0 and B2 or independently in parallel.

For each candidate and/or a subset of the candidates, the following conditions may also be checked: Whether the candidate block has the same motion as the first prediction unit of the current coding unit when the current coding unit is split into two rectangle prediction units and the current prediction unit is the second prediction unit in the coding/decoding order.

Additional conditions related to various properties of current and/or previous slices and/or current and/or neighbour blocks can be utilized for determining whether to include a candidate in the list.

Motion comparison can be realized by comparing a subset of the whole motion information. For example, only the motion vector values for some or all reference picture lists and/or reference indices for some or all reference picture lists and/or an identifier value assigned to each block to represent its motion information can be compared. The comparison can be an identity check or an equivalence check or comparing the (absolute) difference against a threshold or any other similarity metric.

Conditions for deciding whether a candidate is to be included in the list can include motion information comparison with any subset of the candidates as long as not all possible candidate pairs are compared eventually.

Deciding whether a temporal motion vector candidate is to be included in the list can be based on comparing its motion information with motion information of a subset of the spatial motion vector prediction candidates.

When comparing motion information of two blocks, motion information of additional blocks can be considered too. For example, when comparing the block B2 and the block A1 (illustrated in FIG. 12) are checked whether they have the same motion; and when comparing the block B2 and the block B1, all the blocks between the block B2 and the block B1 (illustrated in FIG. 12) are checked whether they have the same motion. This embodiment can be implemented so

that the right-most block of each prediction unit or all blocks of each prediction unit may store the information of how many consecutive blocks to the above have the same motion information. Also the bottom-most block of each prediction unit or all blocks of each prediction unit may store the information of how many consecutive blocks to the left have the same motion information. Using this information the condition for not including B0 in the list can be realized by checking if the number of consecutive blocks with the same motion to the left of B0 is greater than 0. The condition for not including A0 in the list can be realized by checking if the number of consecutive blocks with same motion to the above of A0 is greater than 0. The conditions for not including B2 can be modified as follows:

It is not examined whether the block B2 has same motion as the block B1 or whether the block B2 has same motion as the block A1, but how many consecutive blocks exists to the left of the block B1 with the same motion than the block B1 and/or how many consecutive blocks exist above the block A1 with the same motion. If the number of consecutive blocks with the same motion to the left of the block B1 is greater than the number of blocks between B2 and B1, or if the number of consecutive blocks with the same motion above the block A1 is greater than the number of blocks between the block B2 and the block A1, the block B2 is not included in the merge list.

If the above implementation is used, the value of how many consecutive blocks to the left/above have the same motion information can be determined by direct comparison of motion information or checking the prediction mode and/or the merge index if the block employs a merge process.

When coding/decoding the selected merge index, the information whether the merge process is employed for coding/decoding a Skip mode coding unit or an Inter Merge mode prediction unit can be taken into account. For example, if a context adaptive binary arithmetic coder (CABAC) is used for entropy coding/decoding, different contexts can be used for the bins depending on the coding mode (Skip mode or inter merge mode) of the current block. Furthermore, assigning two contexts depending on whether the merge process is employed in a Skip mode coding unit or an inter Merge mode prediction unit can be applied for only the most significant bin of the merge index.

During the process of removal of redundant candidates, comparison between motion vector predictor candidates can also be based on any other information than the motion vector values. For example, it can be based on linear or non-linear functions of motion vector values, coding or prediction types of the blocks used to obtain the motion information, block size, the spatial location in the frame/ (largest) coding unit/macroblock, the information whether blocks share the same motion with a block, the information whether blocks are in the same coding/prediction unit, etc.

The following pseudo code illustrates an example embodiment of the invention for constructing the merging list.

Inputs to this process are

- a luma location (xP, yP) specifying the top-left luma sample of the current prediction unit relative to the top-left sample of the current picture;
- variables specifying the width and the height of the prediction unit for luma, nPSW and nPSH; and
- a variable PartIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are (with N being replaced by A0, A1, B0, B1 or B2 and with X being replaced by 0 or 1)

19

the availability flags availableFlagN of the neighbouring prediction units,
the reference indices refIdxLXN of the neighbouring prediction units,
the prediction list utilization flags predFlagLXN of the neighbouring prediction units,
the motion vectors mvLXN of the neighbouring prediction units.

For the derivation of availableFlagN, with N being A₀, A₁, B₀, B₁ or B₂ and (xN, yN) being (xP-1, yP+nPSH), (xP-1, yP+nPSH-1), (xP+nPSW, yP-1), (xP+nPSW-1, yP-1) or (xP-1, yP-1), the following applies.

If one of the following conditions is true, the availableFlagN is set equal to 0, both components mvLXN are set equal to 0, refIdxLXN and predFlagLX[xN, yN] of the prediction unit covering luma location (xN, yN) are assigned respectively to mvLXN, refIdxLXN and predFlagLXN.

N is equal to B₂ and availableFlagA₀+availableFlagA₁+availableFlagB₀+availableFlagB₁ is equal to 4.

The prediction unit covering luma location (xN, yN) is not available or PredMode is MODE_INTRA.

N is equal to A₁ and Part Mode of the current prediction unit is PART_Nx2N or PART_nLx2N or PART_nRx2N and PartIdx is equal to 1.

N is equal to A₁ and Part Mode of the current prediction unit is PART_2NxN or PART_2NxN or PART_2NxN and PartIdx is equal to 1 and the prediction units covering luma location (xP+nPSW-1, yP-1) (N=B₁) and luma location (xN, yN) (Cand. N) have identical motion parameters:

mvLX[xP+nPSW-1, yP-1]=mvLX[xN, yN]
refIdxLX[xP+nPSW-1, yP-1]=refIdxLX[xN, yN]
predFlagLX[xP+nPSW-1, yP-1]=predFlagLX[xN, yN]

N is equal to B₁ and Part Mode of the current prediction unit is 2NxN or PART_2NxN or PART_2NxN and PartIdx is equal to 1.

N is equal to B₁ and the prediction units covering luma location (xP-1, yP+nPSH-1) (N=A₁) and luma location (xN, yN) (Cand. N) have identical motion parameters:

mvLX[xP-1, yP+nPSH-1]=mvLX[xN, yN]
refIdxLX[xP-1, yP+nPSH-1]=refIdxLX[xN, yN]
predFlagLX[xP-1, yP+nPSH-1]=predFlagLX[xN, yN]

N is equal to B₀ and the prediction units covering luma location (xP+nPSW-1, yP-1) (N=B₁) and luma location (xN, yN) (Cand. N) have identical motion parameters:

mvLX[xP+nPSW-1, yP-1]=mvLX[xN, yN]
refIdxLX[xP+nPSW-1, yP-1]=refIdxLX[xN, yN]
predFlagLX[xP+nPSW-1, yP-1]=predFlagLX[xN, yN]

N is equal to A₀ and the prediction units covering luma location (xP-1, yP+nPSH-1) (N=A₁) and luma location (xN, yN) (Cand. N) have identical motion parameters:

mvLX[xP-1, yP+nPSH-1]=mvLX[xN, yN]
refIdxLX[xP-1, yP+nPSH-1]=refIdxLX[xN, yN]
predFlagLX[xP-1, yP+nPSH-1]=predFlagLX[xN, yN]

N is equal to B₂ and the prediction units covering luma location (xP+nPSW-1, yP-1) (N=B₁) and luma location (xN, yN) (Cand. N) have identical motion parameters:

20

mvLX[xP+nPSW-1, yP-1]=mvLX[xN, yN]
refIdxLX[xP+nPSW-1, yP-1]=refIdxLX[xN, yN]
predFlagLX[xP+nPSW-1, yP-1]=predFlagLX[xN, yN]

N is equal to B₂ and the prediction units covering luma location (xP-1, yP+nPSH-1) (N=A₁) and luma location (xN, yN) (Cand. N) have identical motion parameters:

mvLX[xP-1, yP+nPSH-1]=mvLX[xN, yN]
refIdxLX[xP-1, yP+nPSH-1]=refIdxLX[xN, yN]
predFlagLX[xP-1, yP+nPSH-1]=predFlagLX[xN, yN]

Part Mode of the current prediction unit is PART_NxN and PartIdx is equal to 3 and the prediction units covering luma location (xP-1, yP) (PartIdx=2) and luma location (xP-1, yP-1) (PartIdx=0) have identical motion parameters:

mvLX[xP-1, yP]=mvLX[xP-1, yP-1]
refIdxLX[xP-1, yP]=refIdxLX[xP-1, yP-1]
predFlagLX[xP-1, yP]=predFlagLX[xP-1, yP-1]

and the prediction units covering luma location (xP, yP-1) (PartIdx=1) and luma location (xN, yN) (Cand. N) have identical motion parameters:

mvLX[xP, yP-1]=mvLX[xN, yN]
refIdxLX[xP, yP-1]=refIdxLX[xN, yN]
predFlagLX[xP, yP-1]=predFlagLX[xN, yN]

Part Mode of the current prediction unit is PART_NxN and PartIdx is equal to 3 and the prediction units covering luma location (xP, yP-1) (PartIdx=1) and luma location (xP-1, yP-1) (PartIdx=0) have identical motion parameters:

mvLX[xP, yP-1]=mvLX[xP-1, yP-1]
refIdxLX[xP, yP-1]=refIdxLX[xP-1, yP-1]
predFlagLX[xP, yP-1]=predFlagLX[xP-1, yP-1]

and the prediction units covering luma location (xP-1, yP) (PartIdx=2) and luma location (xN, yN) (Cand. N) have identical motion parameters:

mvLX[xP-1, yP]=mvLX[xN, yN]
refIdxLX[xP-1, yP]=refIdxLX[xN, yN]
predFlagLX[xP-1, yP]=predFlagLX[xN, yN]

Otherwise, availableFlagN is set equal to 1 and the variables mvLX[xN, yN], refIdxLX[xN, yN] and predFlagLX[xN, yN] of the prediction unit covering luma location (xN, yN) are assigned respectively to mvLXN, refIdxLXN and predFlagLXN.

For the motion vector predictor candidate list generation process, each list candidate can include more information than the motion vector value, such as the reference lists used, the reference frames used in each list and motion vector for each list.

When all motion vector candidates have been examined, one motion vector is selected to be used as the motion vector for the current block. The motion vector selector 364 may examine different motion vectors in the list and determine which motion vector provides the most efficient encoding result, or the selection of the motion vector may be based on to other criteria as well. Information of the selected motion vector is provided for the mode selector for encoding and transmission to the decoder or for storage when the mode selector determines to use inter prediction for the current block. The information may include the index of the motion vector in the list, and/or motion vector parameters or other appropriate information.

The selected motion vector and the block relating to the motion vector is used to generate the prediction representation of the image block 312 which is provided as the output of the mode selector. The output may be used by the first

21

summing device **321** to produce the first prediction error signal **320**, as was described above.

The selected motion vector predictor candidate can be modified by adding a motion vector difference or can be used directly as the motion vector of the block. Moreover, after the motion compensation is performed by using the selected motion vector predictor candidate, the residual signal of the block can be transform coded or skipped to be coded.

Although the embodiments above have been described with respect to the size of the macroblock being 16×16 pixels, it would be appreciated that the methods and apparatus described may be configured to handle macroblocks of different pixel sizes.

In the following the operation of an example embodiment of the decoder **600** is depicted in more detail with reference to FIG. 7.

At the decoder side similar operations are performed to reconstruct the image blocks. FIG. 7 shows a block diagram of a video decoder **700** suitable for employing embodiments of the invention and FIGS. **8a** and **8b** show a flow diagram of an example of a method in the video decoder. The bitstream to be decoded may be received from the encoder, from a network element, from a storage medium or from another source. The decoder is aware of the structure of the bitstream so that it can determine the meaning of the entropy coded codewords and may decode the bitstream by an entropy decoder **701** which performs entropy decoding on the received signal. The entropy decoder thus performs the inverse operation to the entropy encoder **330** of the encoder described above. The entropy decoder **701** outputs the results of the entropy decoding to a prediction error decoder **702** and a pixel predictor **704**.

In some embodiments the entropy coding may not be used but another channel encoding may be in use, or the encoded bitstream may be provided to the decoder **700** without channel encoding. The decoder **700** may comprise a corresponding channel decoder to obtain the encoded codewords from the received signal.

The pixel predictor **704** receives the output of the entropy decoder **701**. The output of the entropy decoder **701** may include an indication on the prediction mode used in encoding the current block. A predictor selector **714** within the pixel predictor **704** determines that an intra-prediction or an inter-prediction is to be carried out. The predictor selector **714** may furthermore output a predicted representation of an image block **716** to a first combiner **713**. The predicted representation of the image block **716** is used in conjunction with the reconstructed prediction error signal **712** to generate a preliminary reconstructed image **718**. The preliminary reconstructed image **718** may be used in the predictor **714** or may be passed to a filter **720**. The filter **720**, if used, applies a filtering which outputs a final reconstructed signal **722**. The final reconstructed signal **722** may be stored in a reference frame memory **724**, the reference frame memory **724** further being connected to the predictor **714** for prediction operations.

Also the prediction error decoder **702** receives the output of the entropy decoder **701**. A dequantizer **792** of the prediction error decoder **702** may dequantize the output of the entropy decoder **701** and the inverse transform block **793** may perform an inverse transform operation to the dequantized signal output by the dequantizer **792**. The output of the entropy decoder **701** may also indicate that prediction error signal is not to be applied and in this case the prediction error decoder produces an all zero output signal.

22

The decoder selects the 16×16 pixel residual macroblock to reconstruct. This residual macroblock is also called as a current block.

The decoder may receive information on the encoding mode used in encoding of the current block. The indication is decoded, when necessary, and provided to the reconstruction processor **791** of the prediction selector **714**. The reconstruction processor **791** examines the indication and selects one of the intra-prediction mode(s), if the indication indicates that the block has been encoded using intra-prediction, or the inter-prediction mode, if the indication indicates that the block has been encoded using inter-prediction.

For inter-prediction mode the reconstruction processor **791** may comprise one or more elements corresponding to the prediction processor **362** of the encoder, such as a motion vector definer, a prediction list modifier and/or a motion vector selector.

The reconstruction processor **791** initializes a motion vector prediction list to default values in block **800**. As was the case in the encoding part, in this example the spatial motion prediction candidates are the spatial neighbour blocks **A0**, **A1**, **B0**, **B1**, **B2** and these spatial motion prediction candidates are processed in the same predetermined order than in the encoder: **A1**, **B1**, **B0**, **A0** and **B2**. The first spatial motion prediction candidate to be selected for further examination is thus **A1**. Before further examination is performed for the selected spatial motion prediction candidate, it is examined whether the merge list already contains a maximum number of spatial motion prediction candidates. If the number of spatial motion prediction candidates in the merge list is not less than the maximum number, the selected spatial motion prediction candidate is not included in the merge list and the process of constructing the merge list can be stopped **826**. On the other hand, if the number of spatial motion prediction candidates in the merge list is less than the maximum number, a further analyses of the selected spatial motion prediction candidate is performed (blocks **804-822**).

The decoder examines **804** if the prediction unit or block covering the spatial motion prediction candidate block is not available for motion prediction. If so, the candidate is not included in the merge list. The reason that the block is not available may be that the block is either coded in intra mode or resides in a different slice or outside of the picture area.

In addition to the common conditions above, for each spatial motion prediction candidate, if any of the following conditions holds, then the candidate is not included in the merge list, otherwise, it is included.

The decoder determines **806** which spatial motion prediction candidate of the set of spatial motion prediction candidates is in question. If the spatial motion prediction candidate is the block **A1**, one or more of the following conditions may be examined **808**, **810** to determine whether to include this spatial motion prediction candidate in the merge list or not. If the current coding unit **100** is vertically split into two rectangle prediction units **103**, **104** as depicted in FIG. **10b** and the current prediction unit is the second prediction unit **104** in the coding/decoding order (**808**), this spatial motion prediction candidate is not included in the merge list. If the current coding unit **100** is not vertically split into two rectangle prediction units but it is horizontally split into two rectangle prediction units **101**, **102** as depicted in FIG. **10a** and the current prediction unit is the second prediction unit in the coding/decoding order and the block **A1** has the same motion information as the block **B1** (**810**), this spatial motion prediction candidate (block **A1**) is not included in the merge list. In the example of FIG. **10a** the

23

second prediction unit is the lower prediction unit **102** of the coding unit **100** and in the example of FIG. **10b** the second prediction unit is the rightmost prediction unit **104** of the coding unit **100**. If none of the conditions above is fulfilled the block **A1** is included in the merge list as a spatial motion prediction candidate (**824**).

If the spatial motion prediction candidate is the block **B1**, one or more of the following conditions may be examined **812**, **814** to determine whether to include this spatial motion prediction candidate in the merge list or not. If the current coding unit **100** is horizontally split into two rectangle prediction units **101**, **102** as depicted in FIG. **10a** and the current prediction unit is the second prediction unit **104** in the coding/decoding order (**812**), this spatial motion prediction candidate is not included in the merge list. If the current coding unit **100** is not horizontally split into two rectangle prediction units and if the block **B1** has the same motion information than the block **A1** (**814**), this spatial motion prediction candidate (block **B1**) is not included in the merge list. If none of the conditions above is fulfilled the block **B1** is included in the merge list as a spatial motion prediction candidate (**824**).

If the spatial motion prediction candidate is the block **B0**, this spatial motion prediction candidate is not included in the merge list if the block **B0** has the same motion information than the block **B1** (**816**). Otherwise, if the number of spatial motion prediction candidates in the merge list is less than the maximum number of spatial motion prediction candidates, this spatial motion prediction candidate (block **B0**) is included in the merge list (**824**).

If the spatial motion prediction candidate is the block **A0**, this spatial motion prediction candidate is not included in the merge list if the block **A0** has the same motion information than the block **A1** (**818**). Otherwise, if the number of spatial motion prediction candidates in the merge list is less than the maximum number of spatial motion prediction candidates, this spatial motion prediction candidate (block **A0**) is included in the merge list (**824**).

If the spatial motion prediction candidate is the block **B2**, this spatial motion prediction candidate is not included in the merge list if the maximum number of spatial motion prediction candidates is four and the other blocks **A0**, **A1**, **B0**, and **B1** are all decided to be included in the merge list (**820**). Otherwise, if the number of spatial motion prediction candidates in the merge list is less than the maximum number of spatial motion prediction candidates, the block **B2** is not included in the merge list if the block **B2** has the same motion information than the block **B1** or the block **A1** (**822**).

Then, after processing the blocks **A1**, **B1**, **B0**, **A0** and **B2** and including a subset of them in the merge list based on the above described conditions, no more redundancy check between these candidates are performed and remaining temporal motion prediction candidate and/or other possible additional candidates may be processed.

When the merge list has been constructed the decoder may use **828** the indication of the motion vector received from the encoder to select the motion vector for decoding the current block. The indication may be, for example, an index to the merge list.

Basically, after the reconstruction processor **791** has constructed the merge list, it would correspond with the merge list constructed by the encoder if the reconstruction processor **791** has the same information available than the encoder had. If some information has been lost during transmission the information from the encoder to the decoder, it may affect the generation of the merge list in the decoder **700**.

24

The above examples describe the operation mainly in the merge mode but the encoder and decoder may also operate in other modes.

The embodiments of the invention described above describe the codec in terms of separate encoder and decoder apparatus in order to assist the understanding of the processes involved. However, it would be appreciated that the apparatus, structures and operations may be implemented as a single encoder-decoder apparatus/structure/operation. Furthermore in some embodiments of the invention the coder and decoder may share some or all common elements.

Although the above examples describe embodiments of the invention operating within a codec within an electronic device, it would be appreciated that the invention as described below may be implemented as part of any video codec. Thus, for example, embodiments of the invention may be implemented in a video codec which may implement video coding over fixed or wired communication paths.

Thus, user equipment may comprise a video codec such as those described in embodiments of the invention above.

It shall be appreciated that the term user equipment is intended to cover any suitable type of wireless user equipment, such as mobile telephones, portable data processing devices or portable web browsers.

Furthermore elements of a public land mobile network (PLMN) may also comprise video codecs as described above.

In general, the various embodiments of the invention may be implemented in hardware or special purpose circuits, software, logic or any combination thereof. For example, some aspects may be implemented in hardware, while other aspects may be implemented in firmware or software which may be executed by a controller, microprocessor or other computing device, although the invention is not limited thereto. While various aspects of the invention may be illustrated and described as block diagrams, flow charts, or using some other pictorial representation, it is well understood that these blocks, apparatus, systems, techniques or methods described herein may be implemented in, as non-limiting examples, hardware, software, firmware, special purpose circuits or logic, general purpose hardware or controller or other computing devices, or some combination thereof.

The embodiments of this invention may be implemented by computer software executable by a data processor of the mobile device, such as in the processor entity, or by hardware, or by a combination of software and hardware. Further in this regard it should be noted that any blocks of the logic flow as in the Figures may represent program steps, or interconnected logic circuits, blocks and functions, or a combination of program steps and logic circuits, blocks and functions. The software may be stored on such physical media as memory chips, or memory blocks implemented within the processor, magnetic media such as hard disk or floppy disks, and optical media such as for example DVD and the data variants thereof, CD.

The memory may be of any type suitable to the local technical environment and may be implemented using any suitable data storage technology, such as semiconductor based memory devices, magnetic memory devices and systems, optical memory devices and systems, fixed memory and removable memory. The data processors may be of any type suitable to the local technical environment, and may include one or more of general purpose computers, special purpose computers, microprocessors, digital signal processors (DSPs) and processors based on multi core processor architecture, as non limiting examples.

25

Embodiments of the inventions may be practiced in various components such as integrated circuit modules. The design of integrated circuits is by and large a highly automated process. Complex and powerful software tools are available for converting a logic level design into a semiconductor circuit design ready to be etched and formed on a semiconductor substrate.

Programs, such as those provided by Synopsys, Inc. of Mountain View, Calif. and Cadence Design, of San Jose, Calif. automatically route conductors and locate components on a semiconductor chip using well established rules of design as well as libraries of pre stored design modules. Once the design for a semiconductor circuit has been completed, the resultant design, in a standardized electronic format (e.g., Opus, GDSII, or the like) may be transmitted to a semiconductor fabrication facility or "fab" for fabrication.

The foregoing description has provided by way of exemplary and non-limiting examples a full and informative description of the exemplary embodiment of this invention. However, various modifications and adaptations may become apparent to those skilled in the relevant arts in view of the foregoing description, when read in conjunction with the accompanying drawings and the appended claims. However, all such and similar modifications of the teachings of this invention will still fall within the scope of this invention.

In the following some examples will be provided.

In some embodiments a method comprises:

receiving a block of pixels including a prediction unit; determining a set of spatial motion vector prediction candidates for the block of pixels; the spatial motion vector prediction candidates being provided with motion information;

selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit;

determining a subset of spatial motion vector predictions based on the location of the block associated with the first spatial motion vector prediction candidate;

comparing motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates;

if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list.

In some embodiments the method comprises including neighbouring blocks of the received block of pixels in the set of spatial motion vector prediction candidates.

In some embodiments the method comprises constructing the set of spatial motion vector predictions by using motion vectors of one or more encoded blocks in a same frame than the block of pixels.

In some embodiments the method comprises selecting spatial motion vector prediction candidates from the set of spatial motion vector prediction candidates as the potential spatial motion vector prediction candidate in a predetermined order.

In some embodiments the method comprises comparing motion information of the potential spatial motion vector prediction candidate with motion information of at most one other spatial motion vector prediction candidate of the set of spatial motion vector prediction candidates.

26

In some embodiments the method comprises prediction unit and a second prediction unit; and if so, excluding the potential spatial motion vector prediction candidate from the merge list if the prediction unit is the second prediction unit.

In some embodiments the method comprises

determining a maximum number of spatial motion vector prediction candidates to be included in a merge list; and limiting the number of spatial motion vector prediction candidates in the merge list smaller or equal to the maximum number.

In some embodiments the method comprises

examining, if the number of spatial motion vector prediction candidates in the merge list smaller than the maximum number;

if so, examining whether a prediction unit to which the potential spatial motion vector prediction candidate belongs is available for motion prediction;

if so, performing at least one of the following:

for the potential spatial motion vector prediction candidate on the left side of the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

the received block of pixels is vertically divided into a first prediction unit and a second prediction unit, and the prediction unit is the second prediction unit;

the received block of pixels is horizontally divided into a first prediction unit and a second prediction unit, and if the prediction unit is the second prediction unit, and the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate above the prediction unit;

for the potential spatial motion vector prediction candidate above the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

the received block of pixels is horizontally divided into a first prediction unit and a second prediction unit, and the prediction unit is the second prediction unit;

the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit;

for the potential spatial motion vector prediction candidate, which is on the right side of the potential spatial motion vector prediction candidate above the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate above the prediction unit;

for the potential spatial motion vector prediction candidate, which is below the potential spatial motion vector prediction candidate on the left side of the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit;

for the potential spatial motion vector prediction candidate cornerwise neighbouring the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled: all the other potential spatial motion vector prediction candidates have been included in the merge list;

27

the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate above the prediction unit;

the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit.

In some embodiments the method comprises including a temporal motion prediction candidate into the merge list.

In some embodiments the method comprises selecting one motion vector prediction candidate from the merge list to represent a motion vector prediction for the block of pixels.

In some embodiments a method according to the second aspect comprises:

receiving an encoded block of pixels including a prediction unit;

determining a set of spatial motion vector prediction candidates for the encoded block of pixels; the spatial motion vector prediction candidates being provided with motion information;

selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit;

determining a subset of spatial motion vector predictions based on the location of the block associated with the first spatial motion vector prediction candidate;

comparing motion information of the first spatial motion vector prediction candidate with motion information of another spatial motion vector prediction candidate of the set of spatial motion vector prediction candidates;

if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list.

In some embodiments the method comprises including neighbouring blocks of the received encoded block of pixels in the set of spatial motion vector prediction candidates.

In some embodiments the method comprises constructing the set of spatial motion vector predictions by using motion vectors of one or more decoded blocks in a same frame than the received encoded block of pixels.

In some embodiments the method comprises selecting spatial motion vector prediction candidates from the set of spatial motion vector prediction candidates as the potential spatial motion vector prediction candidate in a predetermined order.

In some embodiments the method comprises comparing motion information of the potential spatial motion vector prediction candidate with motion information of at most one other spatial motion vector prediction candidate of the set of spatial motion vector prediction candidates.

In some embodiments the method comprises examining whether the received encoded block of pixels is divided into a first prediction unit and a second prediction unit; and if so, excluding the potential spatial motion vector prediction candidate from the merge list if the prediction unit is the second prediction unit.

In some embodiments the method comprises determining a maximum number of spatial motion vector prediction candidates to be included in a merge list; and limiting the number of spatial motion vector prediction candidates in the merge list smaller or equal to the maximum number.

28

In some embodiments the method comprises

examining, if the number of spatial motion vector prediction candidates in the merge list smaller than the maximum number;

if so, examining whether a prediction unit to which the potential spatial motion vector prediction candidate belongs is available for motion prediction;

if so, performing at least one of the following:

for the potential spatial motion vector prediction candidate on the left side of the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

the received encoded block of pixels is vertically divided into a first prediction unit and a second prediction unit, and the prediction unit is the second prediction unit;

the received encoded block of pixels is horizontally divided into a first prediction unit and a second prediction unit, and if the prediction unit is the second prediction unit, and the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate above the prediction unit;

for the potential spatial motion vector prediction candidate above the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

the received encoded block of pixels is horizontally divided into a first prediction unit and a second prediction unit, and the prediction unit is the second prediction unit;

the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit;

for the potential spatial motion vector prediction candidate, which is on the right side of the potential spatial motion vector prediction candidate above the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate above the prediction unit;

for the potential spatial motion vector prediction candidate, which is below the potential spatial motion vector prediction candidate on the left side of the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit;

for the potential spatial motion vector prediction candidate cornerwise neighbouring the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

all the other potential spatial motion vector prediction candidates have been included in the merge list;

the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate above the prediction unit;

the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit.

In some embodiments the method comprises including a temporal motion prediction candidate into the merge list.

29

In some embodiments the method comprises selecting one motion vector prediction candidate from the merge list to represent a motion vector prediction for the received encoded block of pixels.

In some embodiments an apparatus according to the third aspect comprises a processor and a memory including computer program code, the memory and the computer program code configured to, with the processor, cause the apparatus to:

receive a block of pixels

including a prediction unit;

determining a set of spatial motion vector prediction candidates for the block of pixels; the spatial motion vector prediction candidates being provided with motion information;

selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit;

determining a subset of spatial motion vector predictions based on the location of the block associated with the first spatial motion vector prediction candidate;

comparing motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates;

if at least one the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list.

In some embodiments an apparatus according to the fourth aspect comprises a processor and a memory including computer program code, the memory and the computer program code configured to, with the processor, cause the apparatus to:

receive an encoded block of pixels including a prediction unit;

determine a set of spatial motion vector prediction candidates for the encoded block of pixels; the spatial motion vector prediction candidates being provided with motion information;

select a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit;

determine a subset of spatial motion vector predictions based on the location of the block associated with the first spatial motion vector prediction candidate;

compare motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates;

exclude the first spatial motion vector prediction candidate from the merge list, if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other

In some embodiments a storage medium having stored thereon a computer program code a computer executable program code for use by an encoder, said program codes comprise instructions for use by an encoder, said program code comprises instructions for:

30

receiving a block of pixels including a prediction unit; determining a set of spatial motion vector prediction candidates for the block of pixels; the spatial motion vector prediction candidates being provided with motion information;

selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit;

determining a subset of spatial motion vector predictions based on the location of the block associated with the first spatial motion vector prediction candidate;

comparing motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates;

if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list.

In some embodiments a storage medium having stored thereon a computer program code a computer executable program code for use by an encoder, said program codes comprise instructions for use by an encoder, said program code comprises instructions for:

receiving an encoded block of pixels including a prediction unit;

determining a set of spatial motion vector prediction candidates for the encoded block of pixels; the spatial motion vector prediction candidates being provided with motion information;

selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit;

determining a subset of spatial motion vector predictions based on the location of the block associated with the first spatial motion vector prediction candidate;

comparing motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates;

if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list.

In some embodiments an apparatus comprises:

means for receiving a block of pixels including a prediction unit;

means for selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit;

means for determining a subset of spatial motion vector predictions based on the location of the block associated with the first spatial motion vector prediction candidate;

means for comparing motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates;

means for excluding the first spatial motion vector prediction candidate from the merge list, if at least one of the

31

comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other.

In some embodiments an apparatus comprises:

means for receiving an encoded block of pixels including a prediction unit;

means for determining a set of spatial motion vector prediction candidates for the encoded block of pixels; the spatial motion vector prediction candidates being provided with motion information;

means for selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit;

means for determining a subset of spatial motion vector predictions based on the location of the block associated with the first spatial motion vector prediction candidate;

means for comparing motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates;

means for excluding the first spatial motion vector prediction candidate from the merge list, if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other.

The invention claimed is:

1. A method comprising:

receiving a block of pixels including a prediction unit; determining a set of spatial motion vector prediction candidates for the block of pixels; the spatial motion vector prediction candidates being provided with motion information;

selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit, where the merge list is constructed based on the motion information of the spatial motion vector prediction candidates and is utilized to identify motion vector prediction candidates of which one spatial motion vector prediction candidate from the merge list is signaled as the motion information for the prediction unit;

determining a subset of spatial motion vector prediction candidates based on a location of the block associated with the first spatial motion vector prediction candidate;

comparing motion information of the first spatial motion vector prediction candidate with motion information of spatial motion vector prediction candidates in the determined subset of spatial motion vector prediction candidates without making a comparison of each possible candidate pair from the set of spatial motion vector prediction candidates, wherein comparing comprises performing an equivalence check or comparing a difference in motion information to a threshold or other similarity metric;

if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list; and

32

causing information identifying the one spatial motion vector prediction candidate from the merge list to be transmitted to a decoder or to be stored.

2. The method according to claim 1 comprising selecting spatial motion vector prediction candidates from the set of spatial motion vector prediction candidates as the potential spatial motion vector prediction candidate in a predetermined order.

3. The method according to claim 1, comprising comparing motion information of the potential spatial motion vector prediction candidate with motion information of at most one other spatial motion vector prediction candidate of the set of spatial motion vector prediction candidates.

4. The method according to claim 1 comprising examining whether the received block of pixels is divided into a first prediction unit and a second prediction unit; and if so, excluding the potential spatial motion vector prediction candidate from the merge list if the prediction unit is the second prediction unit.

5. The method according to claim 1, further comprising determining a maximum number of spatial motion vector prediction candidates to be included in a merge list; and limiting the number of spatial motion vector prediction candidates in the merge list smaller or equal to the maximum number.

6. The method according to claim 5 comprising:

examining, if the number of spatial motion vector prediction candidates in the merge list smaller than the maximum number;

if so, examining whether the prediction unit to which the potential spatial motion vector prediction candidate belongs is available for motion prediction;

if so, performing at least one of the following:

for a potential spatial motion vector prediction candidate on a left side of the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

the received block of pixels is vertically divided into a first prediction unit and a second prediction unit;

the received block of pixels is horizontally divided into a first prediction unit and a second prediction unit,

and if the prediction unit is the second prediction unit, and the potential spatial motion vector prediction candidate has essentially similar motion information than a spatial motion vector prediction candidate above the prediction unit;

for a potential spatial motion vector prediction candidate above the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

the received block of pixels is horizontally divided into a first prediction unit and a second prediction unit,

and the prediction unit is the second prediction unit; the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit;

for a potential spatial motion vector prediction candidate, which is on a right side of the potential spatial motion vector prediction candidate above the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate above the prediction unit;

33

for a potential spatial motion vector prediction candidate, which is below the potential spatial motion vector prediction candidate on the left side of the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit;

for a potential spatial motion vector prediction candidate cornerwise neighbouring the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

- all the other potential spatial motion vector prediction candidates have been included in the merge list;
- the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate above the prediction unit;
- the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit.

7. The method according to claim 1 further comprising including a temporal motion prediction candidate into the merge list.

8. The method according to claim 1 comprising selecting one motion vector prediction candidate from the merge list to represent a motion vector prediction for the block of pixels.

9. A method comprising:

- receiving an encoded block of pixels including a prediction unit and information identifying a respective spatial motion vector prediction candidate from a merge list constructed by an encoder;
- determining a set of spatial motion vector prediction candidates for the encoded block of pixels; the spatial motion vector prediction candidates being provided with motion information;
- selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit, where the merge list is constructed based on the motion information of the spatial motion vector prediction candidates;
- determining a subset of spatial motion vector prediction candidates based on the location of the block associated with the first spatial motion vector prediction candidate;
- comparing motion information of the first spatial motion vector prediction candidate with motion information of another spatial motion vector prediction candidate of the set of spatial motion vector prediction candidates without making a comparison of each possible candidate pair from the set of spatial motion vector prediction candidates, wherein comparing comprises performing an equivalence check or comparing a difference in motion information to a threshold or other similarity metric;
- if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list; and

34

selecting a spatial motion vector prediction candidate from the merge list for use in decoding the encoded block of pixels, wherein the spatial motion vector prediction candidate is selected from the merge list using the information that was received identifying a respective spatial motion vector prediction candidate.

10. The method according to claim 9 comprising comparing motion information of the potential spatial motion vector prediction candidate with motion information of at most one other spatial motion vector prediction candidate of the set of spatial motion vector prediction candidates.

11. The method according to claim 9 comprising examining whether the received encoded block of pixels is divided into a first prediction unit and a second prediction unit; and if so, excluding the potential spatial motion vector prediction candidate from the merge list if the prediction unit is the second prediction unit.

12. The method according to claim 9 further comprising determining a maximum number of spatial motion vector prediction candidates to be included in a merge list; and limiting the number of spatial motion vector prediction candidates in the merge list smaller or equal to the maximum number.

13. The method according to claim 12 comprising: examining, if the number of spatial motion vector prediction candidates in the merge list smaller than the maximum number;

if so, examining whether the prediction unit to which the potential spatial motion vector prediction candidate belongs is available for motion prediction;

if so, performing at least one of the following:

for a potential spatial motion vector prediction candidate on a left side of the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

- the received encoded block of pixels is vertically divided into a first prediction unit and a second prediction unit;

- the received encoded block of pixels is horizontally divided into a first prediction unit and a second prediction unit, and if the prediction unit is the second prediction unit, and the potential spatial motion vector prediction candidate has essentially similar motion information than a spatial motion vector prediction candidate above the prediction unit;

for a potential spatial motion vector prediction candidate above the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

- the received encoded block of pixels is horizontally divided into a first prediction unit and a second prediction unit, and the prediction unit is the second prediction unit;

- the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit;

for a potential spatial motion vector prediction candidate, which is on a right side of the potential spatial motion vector prediction candidate above the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate above the prediction unit;

35

for a potential spatial motion vector prediction candidate, which is below the potential spatial motion vector prediction candidate on the left side of the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit; and

for a potential spatial motion vector prediction candidate cornerwise neighbouring the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

all the other potential spatial motion vector prediction candidates have been included in the merge list;

the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate above the prediction unit;

the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit.

14. The method according to claim 9 comprising selecting one motion vector prediction candidate from the merge list to represent a motion vector prediction for the received encoded block of pixels.

15. An apparatus comprising a processor and a memory including computer program code, the memory and the computer program code configured to, with the processor, cause the apparatus to:

receive a block of pixels including a prediction unit;

determine a set of spatial motion vector prediction candidates for the block of pixels; the spatial motion vector prediction candidates being provided with motion information;

select a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit, where the merge list is constructed based on the motion information of the spatial motion vector prediction candidates and is utilized to identify motion vector prediction candidates of which one spatial motion vector prediction candidate from the merge list is signaled as the motion information for the prediction unit;

determine a subset of spatial motion vector-prediction candidates based on the location of the block associated with the first spatial motion vector prediction candidate;

compare motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates without making a comparison of each possible candidate pair from the set of spatial motion vector prediction candidates, wherein comparing comprises performing an equivalence check or comparing a difference in motion information to a threshold or other similarity metric;

if at least one the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, exclude the first spatial motion vector prediction candidate from the merge list; and

36

cause information identifying the one spatial motion vector prediction candidate from the merge list to be transmitted to a decoder or to be stored.

16. An apparatus comprising a processor and a memory including computer program code, the memory and the computer program code configured to, with the processor, cause the apparatus to:

receive an encoded block of pixels including a prediction unit and information identifying a respective spatial motion vector prediction candidate from a merge list constructed by an encoder;

determine a set of spatial motion vector prediction candidates for the encoded block of pixels; the spatial motion vector prediction candidates being provided with motion information;

select a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit, where the merge list is constructed based on the motion information of the spatial motion vector prediction candidates;

determine a subset of spatial motion vector prediction candidates based on the location of the block associated with the first spatial motion vector prediction candidate;

compare motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates without making a comparison of each possible candidate pair from the set of spatial motion vector prediction candidates, wherein comparing comprises performing an equivalence check or comparing a difference in motion information to a threshold or other similarity metric;

exclude the first spatial motion vector prediction candidate from the merge list, if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other; and

select a spatial motion vector prediction candidate from the merge list for use in decoding the encoded block of pixels, wherein the spatial motion vector prediction candidate is selected from the merge list using the information that was received identifying a respective spatial motion vector prediction candidate.

17. A non-transitory computer readable medium having stored thereon a computer executable program code for use by an encoder, said program codes comprise instructions for use by an encoder, said program code comprises instructions for:

receiving a block of pixels including a prediction unit; determining a set of spatial motion vector prediction candidates for the block of pixels; the spatial motion vector prediction candidates being provided with motion information;

selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit, where the merge list is constructed based on the motion information of the spatial motion vector prediction candidates and is utilized to identify motion vector prediction candidates of which one spatial

37

motion vector prediction candidate from the merge list is signaled as the motion information for the prediction unit;

determining a subset of spatial motion vector prediction candidates based on the location of the block associated with the first spatial motion vector prediction candidate;

comparing motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates without making a comparison of each possible candidate pair from the set of spatial motion vector prediction candidates, wherein comparing comprises performing an equivalence check or comparing a difference in motion information to a threshold or other similarity metric;

if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list; and

causing information identifying the one spatial motion vector prediction candidate from the merge list to be transmitted to a decoder or to be stored.

18. A non-transitory computer readable medium having stored thereon a computer executable program code for use by an encoder, said program codes comprise instructions for use by an encoder, said program code comprises instructions for:

receiving an encoded block of pixels including a prediction unit and information identifying a respective spatial motion vector prediction candidate from a merge list constructed by an encoder;

determining a set of spatial motion vector prediction candidates for the encoded block of pixels; the spatial motion vector prediction candidates being provided with motion information;

selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit, where the merge list is constructed based on the motion information of the spatial motion vector prediction candidates;

determining a subset of spatial motion vector prediction candidates based on the location of the block associated with the first spatial motion vector prediction candidate;

comparing motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates without making a comparison of each possible candidate pair from the set of spatial motion vector prediction candidates, wherein comparing comprises performing an equivalence check or comparing a difference in motion information to a threshold or other similarity metric;

if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list; and

selecting a spatial motion vector prediction candidate from the merge list for use in decoding the encoded block of pixels, wherein the spatial motion vector

38

prediction candidate is selected from the merge list using the information that was received identifying a respective spatial motion vector prediction candidate.

19. The apparatus according to claim 15 wherein the apparatus is further caused to select spatial motion vector prediction candidates from the set of spatial motion vector prediction candidates as the potential spatial motion vector prediction candidate in a predetermined order.

20. The apparatus according to claim 15, wherein the apparatus is further caused to examine whether the received block of pixels is divided into a first prediction unit and a second prediction unit; and if so, exclude the potential spatial motion vector prediction candidate from the merge list if the prediction unit is the second prediction unit.

21. The apparatus according to claim 15 wherein the apparatus is further caused to examine whether the received block of pixels is divided into a first prediction unit and a second prediction unit; and if so, exclude the potential spatial motion vector prediction candidate from the merge list if the prediction unit is the second prediction unit.

22. The apparatus according to claim 15, wherein the apparatus is further caused to:

determine a maximum number of spatial motion vector prediction candidates to be included in a merge list; and

limit the number of spatial motion vector prediction candidates in the merge list smaller or equal to the maximum number.

23. The apparatus according to claim 22 wherein the apparatus is further caused to:

examine, if the number of spatial motion vector prediction candidates in the merge list smaller than the maximum number;

if so, examine whether the prediction unit to which the potential spatial motion vector prediction candidate belongs is available for motion prediction;

if so, perform at least one of the following:

for a potential spatial motion vector prediction candidate on a left side of the prediction unit, exclude the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

the received block of pixels is vertically divided into a first prediction unit and a second prediction unit;

the received block of pixels is horizontally divided into a first prediction unit and a second prediction unit, and if the prediction unit is the second prediction unit, and the potential spatial motion vector prediction candidate has essentially similar motion information than a spatial motion vector prediction candidate above the prediction unit;

for a potential spatial motion vector prediction candidate above the prediction unit, exclude the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

the received block of pixels is horizontally divided into a first prediction unit and a second prediction unit, and the prediction unit is the second prediction unit;

the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit;

for a potential spatial motion vector prediction candidate, which is on a right side of the potential spatial motion vector prediction candidate above the prediction unit, exclude the potential spatial motion vector prediction candidate from the merge list if the potential spatial motion vector prediction candidate has essentially

39

similar motion information than the spatial motion vector prediction candidate above the prediction unit; for a potential spatial motion vector prediction candidate, which is below the potential spatial motion vector prediction candidate on the left side of the prediction unit, exclude the potential spatial motion vector prediction candidate from the merge list if the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit;

for a potential spatial motion vector prediction candidate cornerwise neighbouring the prediction unit, exclude the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

- all the other potential spatial motion vector prediction candidates have been included in the merge list;
- the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate above the prediction unit;
- the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit.

24. The apparatus according to claim 15 wherein the apparatus is further caused to include a temporal motion prediction candidate into the merge list.

25. The apparatus according to claim 15 wherein the apparatus is further caused to select one motion vector prediction candidate from the merge list to represent a motion vector prediction for the block of pixels.

26. The apparatus according to claim 16 wherein the apparatus is further caused to compare motion information of the potential spatial motion vector prediction candidate with motion information of at most one other spatial motion vector prediction candidate of the set of spatial motion vector prediction candidates.

27. The apparatus according to claim 16 wherein the apparatus is further caused to examine whether the received encoded block of pixels is divided into a first prediction unit and a second prediction unit; and if so, exclude the potential spatial motion vector prediction candidate from the merge list if the prediction unit is the second prediction unit.

28. The apparatus according to claim 16 wherein the apparatus is further caused to:

- determine a maximum number of spatial motion vector prediction candidates to be included in a merge list; and
- limit the number of spatial motion vector prediction candidates in the merge list smaller or equal to the maximum number.

29. The apparatus according to claim 28 wherein the apparatus is further caused to:

- examine if the number of spatial motion vector prediction candidates in the merge list smaller than the maximum number;

if so, examine whether the prediction unit to which the potential spatial motion vector prediction candidate belongs is available for motion prediction;

if so, perform at least one of the following:

- for a potential spatial motion vector prediction candidate on a left side of the prediction unit, exclude the

40

potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

- the received encoded block of pixels is vertically divided into a first prediction unit and a second prediction unit;

- the received encoded block of pixels is horizontally divided into a first prediction unit and a second prediction unit, and if the prediction unit is the second prediction unit, and the potential spatial motion vector prediction candidate has essentially similar motion information than a spatial motion vector prediction candidate above the prediction unit;

for a potential spatial motion vector prediction candidate above the prediction unit, exclude the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

- the received encoded block of pixels is horizontally divided into a first prediction unit and a second prediction unit, and the prediction unit is the second prediction unit;

- the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit;

for a potential spatial motion vector prediction candidate, which is on a right side of the potential spatial motion vector prediction candidate above the prediction unit, exclude the potential spatial motion vector prediction candidate from the merge list if the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate above the prediction unit;

for a potential spatial motion vector prediction candidate, which is below the potential spatial motion vector prediction candidate on the left side of the prediction unit, exclude the potential spatial motion vector prediction candidate from the merge list if the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit; and

for a potential spatial motion vector prediction candidate cornerwise neighbouring the prediction unit, exclude the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

- all the other potential spatial motion vector prediction candidates have been included in the merge list;

- the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate above the prediction unit;

- the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit.

30. The apparatus according to claim 16 wherein the apparatus is further caused to select one motion vector prediction candidate from the merge list to represent a motion vector prediction for the received encoded block of pixels.

* * * * *

EXHIBIT 17



US011805267B2

(12) **United States Patent**
Ugur et al.

(10) **Patent No.:** **US 11,805,267 B2**
(45) **Date of Patent:** ***Oct. 31, 2023**

(54) **MOTION PREDICTION IN VIDEO CODING**

(71) Applicant: **Nokia Technologies Oy**, Espoo (FI)

(72) Inventors: **Kemal Ugur**, Tampere (FI); **Jani Lainema**, Tampere (FI); **Antti Hallapuro**, Tampere (FI)

(73) Assignee: **NOKIA TECHNOLOGIES OY**, Espoo (FI)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **17/328,750**

(22) Filed: **May 24, 2021**

(65) **Prior Publication Data**

US 2021/0281869 A1 Sep. 9, 2021

Related U.S. Application Data

(63) Continuation of application No. 16/729,974, filed on Dec. 30, 2019, now Pat. No. 11,019,354, which is a (Continued)

(51) **Int. Cl.**
H04N 19/50 (2014.01)
H04N 19/42 (2014.01)
(Continued)

(52) **U.S. Cl.**
CPC **H04N 19/50** (2014.11); **H04N 19/105** (2014.11); **H04N 19/42** (2014.11); **H04N 19/523** (2014.11);
(Continued)

(58) **Field of Classification Search**

None

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,404,815 B1 6/2002 Sekiguchi et al.
6,512,523 B1 1/2003 Gross
(Continued)

FOREIGN PATENT DOCUMENTS

CA 2729615 A1 1/2010
CN 101523922 A 9/2009
(Continued)

OTHER PUBLICATIONS

“Advanced Video Coding for Generic Audiovisual Services”, Series H: Audiovisual And Multimedia Systems, Infrastructure of audiovisual services—Coding of moving video, ITU-T Recommendation H.264, Nov. 2007, 564 pages.

(Continued)

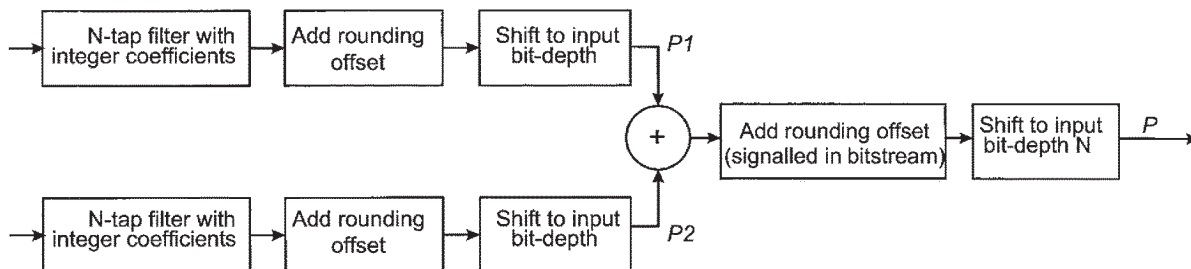
Primary Examiner — Peter D Le

(74) *Attorney, Agent, or Firm* — ALSTON & BIRD LLP

(57) **ABSTRACT**

Apparatuses, methods and computer programs are provided for utilizing motion prediction in video coding. A block of pixels of a video representation encoded in a bitstream is read, and a type of the block is determined. If the determining indicates that the block is a block predicted by using two or more reference blocks, a first reference pixel location in a first reference block is determined and a second reference pixel location in a second reference block is determined. The first reference pixel location is used to obtain a first prediction. The first prediction has a second precision, which is higher than the first precision. The second reference pixel location is used to obtain a second prediction, which also has the second precision. The first prediction and the second prediction are combined to obtain a combined prediction; and the precision of the combined prediction is reduced to the first precision.

36 Claims, 11 Drawing Sheets



Related U.S. Application Data

continuation of application No. 15/876,495, filed on Jan. 22, 2018, now Pat. No. 10,523,960, which is a continuation of application No. 15/490,469, filed on Apr. 18, 2017, now Pat. No. 9,877,037, which is a continuation of application No. 15/250,124, filed on Aug. 29, 2016, now Pat. No. 9,628,816, which is a continuation of application No. 13/344,893, filed on Jan. 6, 2012, now Pat. No. 9,432,693.

- (60) Provisional application No. 61/430,694, filed on Jan. 7, 2011.

(51) Int. Cl.

H04N 19/523 (2014.01)

H04N 19/577 (2014.01)

H04N 19/105 (2014.01)

H04N 19/176 (2014.01)

H04N 19/182 (2014.01)

H04N 19/184 (2014.01)

(52) U.S. Cl.

CPC **H04N 19/577** (2014.11); **H04N 19/176** (2014.11); **H04N 19/182** (2014.11); **H04N 19/184** (2014.11)

(56) References Cited**U.S. PATENT DOCUMENTS**

6,539,058	B1	3/2003	Pearlstein et al.
6,950,469	B2	9/2005	Karczewicz et al.
7,580,456	B2	8/2009	Li et al.
8,005,137	B2	8/2011	Han et al.
8,149,910	B2	4/2012	Tanizawa et al.
8,284,835	B2	10/2012	Iguchi
8,428,133	B2	4/2013	Ye et al.
8,498,336	B2	7/2013	Tourapis et al.
8,660,174	B2	2/2014	Fu et al.
8,676,000	B2	3/2014	Alshina et al.
8,711,939	B2	4/2014	Alshina et al.
8,750,378	B2	6/2014	Karczewicz et al.
8,995,526	B2	3/2015	Karczewicz et al.
9,014,280	B2	4/2015	Ye et al.
9,161,057	B2	10/2015	Karczewicz et al.
9,237,355	B2	1/2016	Chien et al.
9,307,122	B2	4/2016	Ugur et al.
9,432,693	B2	8/2016	Ugur et al.
9,877,037	B2	1/2018	Ugur et al.
2003/0202607	A1	10/2003	Srinivasan
2004/0208247	A1	10/2004	Barrau et al.
2005/0105620	A1	5/2005	Fukushima
2005/0207496	A1	9/2005	Komiya et al.
2008/0075169	A1	3/2008	Ugur et al.
2008/0089417	A1	4/2008	Bao et al.
2009/0087111	A1	4/2009	Noda et al.
2009/0092188	A1	4/2009	Lee et al.
2009/0232215	A1	9/2009	Park et al.
2009/0257499	A1	10/2009	Karczewicz et al.
2009/0257503	A1	10/2009	Ye et al.
2010/0002770	A1	1/2010	Motta et al.
2010/0086027	A1	4/2010	Panchai et al.
2010/0111182	A1	5/2010	Karczewicz et al.
2011/0032991	A1	2/2011	Sekiguchi et al.
2011/0090966	A1	4/2011	Chujoh et al.
2011/0200108	A1	8/2011	Joshi et al.
2012/0051431	A1	3/2012	Chien et al.
2012/0063515	A1	3/2012	Panchai et al.
2013/0142262	A1	6/2013	Ye et al.
2013/0182763	A1	7/2013	Yasuda et al.

FOREIGN PATENT DOCUMENTS

CN	101816183	A	8/2010
RU	2004103743	A	6/2005

RU	2008138706	A	4/2010
WO	WO 2008/048864	A2	4/2008
WO	WO 2008/067734	A1	6/2008
WO	WO 2010/001832	A1	1/2010

OTHER PUBLICATIONS

Alvarez, J., "Discrepancies in Documentation and Implementation of Sub-pel Interpolation in TML-8 (Draft 0)", ITU—Telecommunications Standardization Sector, Study Group 16 Question 6, Video Coding Experts Group (VCEG), 14th Meeting, Santa Barbara, CA (Sep. 24-27, 2001), 10 pages.

Chen et al., "Bidirectional MC-EZBC with Lifting Implementation", IEEE Transactions on Circuits and Systems for Video Technology, vol. 14, Issue 10, (Oct. 2004), 26 pages.

Chiu et al., "Description of Video Coding Technology Proposal: Self Derivation of Motion Estimation and Adaptive (Wiener) Loop Filtering", Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 1st Meeting, (Apr. 15-23, 2010), 29 pages.

Chujoh, T et al., "Bidirectional prediction for stored B-slice," ITU-T SG16 contribution, VCEG-A120, Berlin, Germany, Jul. 2008.

Extended European Search Report from corresponding European Patent Application No. 12731927.5 dated May 6, 2016.

Final Office Action for U.S. Appl. No. 13/344,893 dated Jan. 23, 2015, 20 pages.

International Search Report and Written Opinion received for corresponding Patent Cooperation Treaty Application No. PCT/IB2012/050089 dated May 9, 2012, 14 pages.

Lopez et al., "A Flexible Template for H.264/AVC Block Matching Motion Estimation Architectures" IEEE Transactions on Consumer Electronics, vol. 54, No. 2, (May 2008), 7 pages.

Lopez et al., "A High Quality/Low Computational Cost Technique for Block Matching Motion Estimation", IEEE Computer Society, Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, (Mar. 7-11, 2005), 6 pages.

Non-Final Office Action for U.S. Appl. No. 13/344,893 dated Jul. 16, 2014, 24 pages.

Non-Final Office Action for U.S. Appl. No. 13/344,893 dated Jul. 28, 2015, 17 pages.

Notice of Allowance for U.S. Appl. No. 13/344,893, dated Jun. 14, 2016, 3 pages.

Notice of Allowance for U.S. Appl. No. 15/250,124, dated Feb. 23, 2017, 3 pages.

Notice of Allowance for U.S. Appl. No. 15/490,469, dated Sep. 15, 2017, 13 pages.

Notice of Allowance for U.S. Appl. No. 15/876,495 dated Aug. 23, 2019.

Notice of Allowance for U.S. Appl. No. 16/729,974 dated Jan. 25, 2021.

Notice of Allowance for U.S. Appl. No. 13/344,893 dated Apr. 27, 2016, 20 pages.

Notice of Allowance for U.S. Appl. No. 15/250,124 dated Dec. 13, 2016, 13 pages.

Office Action for Chinese Patent Application No. 2012800096959 dated Mar. 20, 2017, with English summary, 7 pages.

Office Action for European Application No. 12 731 927.5 dated Nov. 20, 2019.

Office Action for India Application No. 6227/CHENP/2013, dated Apr. 2, 2018, 5 pages.

Office Action for U.S. Appl. No. 15/876,495 dated Mar. 8, 2019.

Office Action from Korean Patent Application No. 2013-7020731, dated Aug. 26, 2015.

Office Action from Korean Patent Application No. 2013-7020731, dated Jul. 31, 2014.

Office Action from Russian Patent Application No. 2013136693, dated Nov. 28, 2014.

Office Action from Vietnamese Patent Application No. 1-2013-02120 dated Jan. 23, 2017, with English Language translation, 2 pages.

Summons to Attend Oral Proceedings for European Application No. 12731927.5 dated Oct. 12, 2020, 9 pages.

(56)

References Cited**OTHER PUBLICATIONS**

Toivonen, T., "Efficient Methods for Video Coding and Processing", University of Oulu, Acta Universitatis Ouluensis, C Technica 290, (Dec. 2007), 111 pages.

Tsung et al., "Single-Iteration Full-Search Fractional Motion Estimation for Quad Full HD H.264/AVC Encoding", 2009 IEEE International Conference on Multimedia and Expo, (Jun. 28-Jul. 3, 2009), 4 pages.

Ugur et al., "High precision bi-directional averaging", Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T WP3 and ISO/IEC JTC1/SC29/NVG11, 4th Meeting, Jan. 20-28, 2011, pp. 1-3.

Ugur et al., "On clipping in bi-directional averaging", Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T WP3 and ISO/IEC JTC1/SC29/WG11, 5th Meeting, Mar. 16-23, 2011, pp. 1-4.

Wang et al., "Motion Estimation and Mode Decision for Low-Complexity H.264 Decoder", Tech. Rep. 210-2005-4, Columbia University DVMM Group, (2005), 25 pages.

Yang et al., "Prediction-Based Directional Fractional Pixel Motion Estimation for H.264 Video Coding", IEEE International Conference on Acoustics, Speech, and Signal Processing, (Mar. 23, 2005), 4 pages.

Ye et al., "High Precision Interpolation and Prediction"; 35 VCEG Meeting; 85. MPEG Meeting; Jul. 16, 2008-Jul. 18, 2008; Berlin; Video Coding Experts Group of Itu-T SG.16; No. VCEG-AI33; Jul. 12, 2008; XP030003598.

Yi-Jen Chiu et al.: "TEI: Fast Techniques to Improve Self Derivation of Motion Estimation"; JCT-VC Meeting; Jul. 21, 2010-Jul. 28, 2010; Geneva; Joint Collaborative Team on Video Coding of ISO/IEC JTC1/SC29/WG11 and ITU-T SG.16; Jul. 28, 2010; XP030007627.

Extended European Search Report for European Application No. 22173168.0 dated Oct. 26, 2022, 11 pages.

Decision to Grant for Chinese Application No. 201280009695.9 dated Jun. 21, 2017, 4 pages.

Decision to Grant for Russian Application No. 2013136693/08 dated Jun. 4, 2015, 12 pages.

Minutes of the Oral Proceedings for European Application No. 12731927.5 dated Dec. 9, 2021, 27 pages.

Notice of Allowance for Vietnamese Application No. 1-2013-02120 dated Aug. 31, 2017, 2 pages.

Office Action for Chinese Application No. 201280009695.9 dated Feb. 15, 2016, 10 pages.

Yoshino et al., "Enhanced Switching of Interpolation Filter for HEVC", Joint Collaborative Team on Video Coding (JCT-VC) or ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 3rd Meeting, JCTVC-C183, (Oct. 7-15, 2010), 4 pages.

Office Action for European Application No. 22173168.0 dated Sep. 11, 2023, 6 pages.

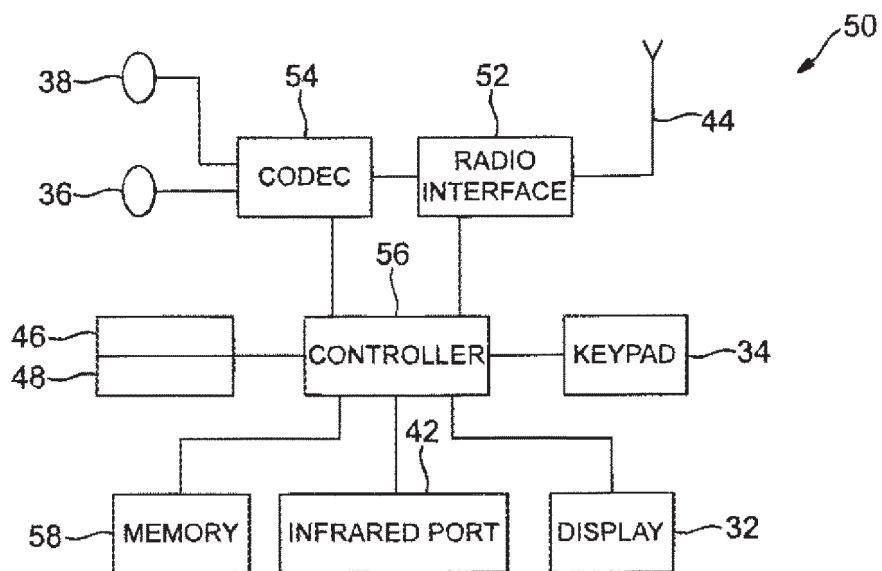


FIG. 1

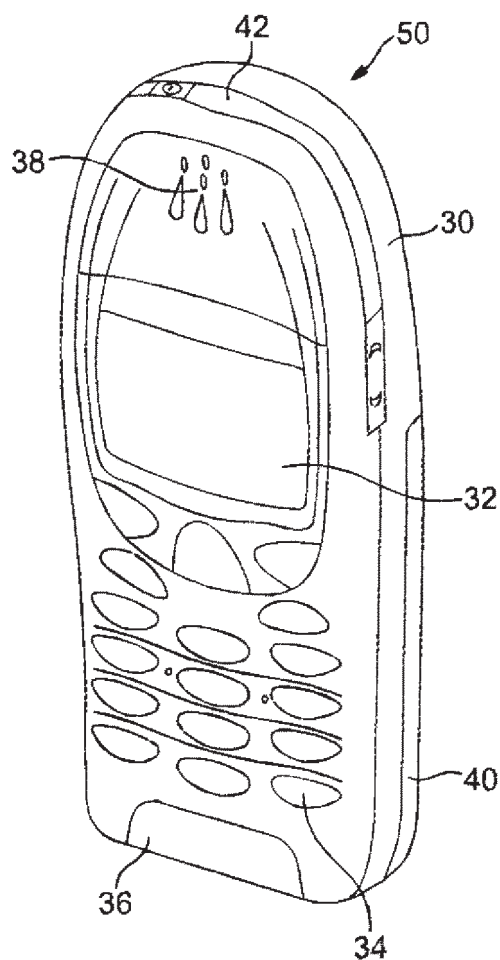


FIG. 2

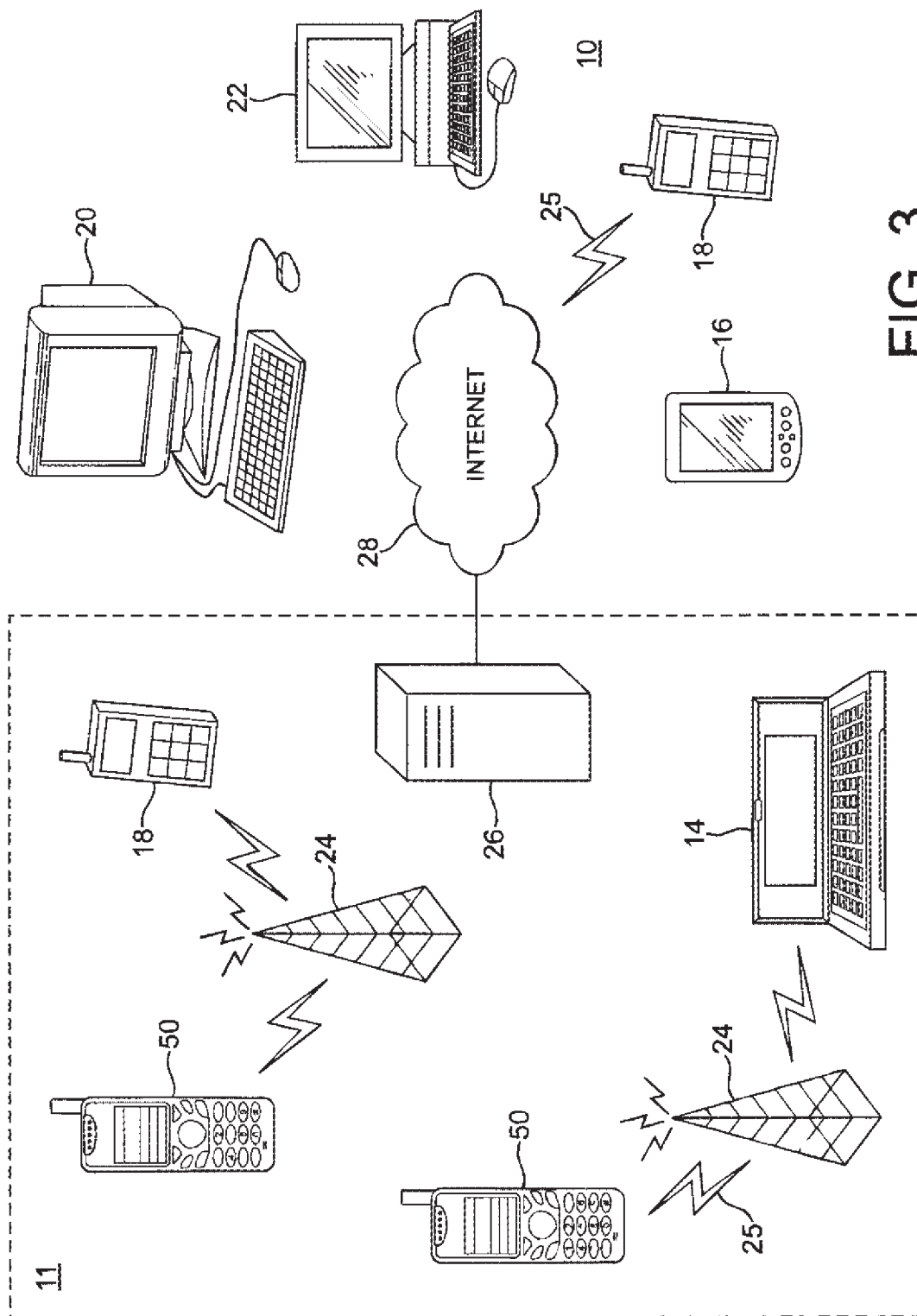


FIG. 3

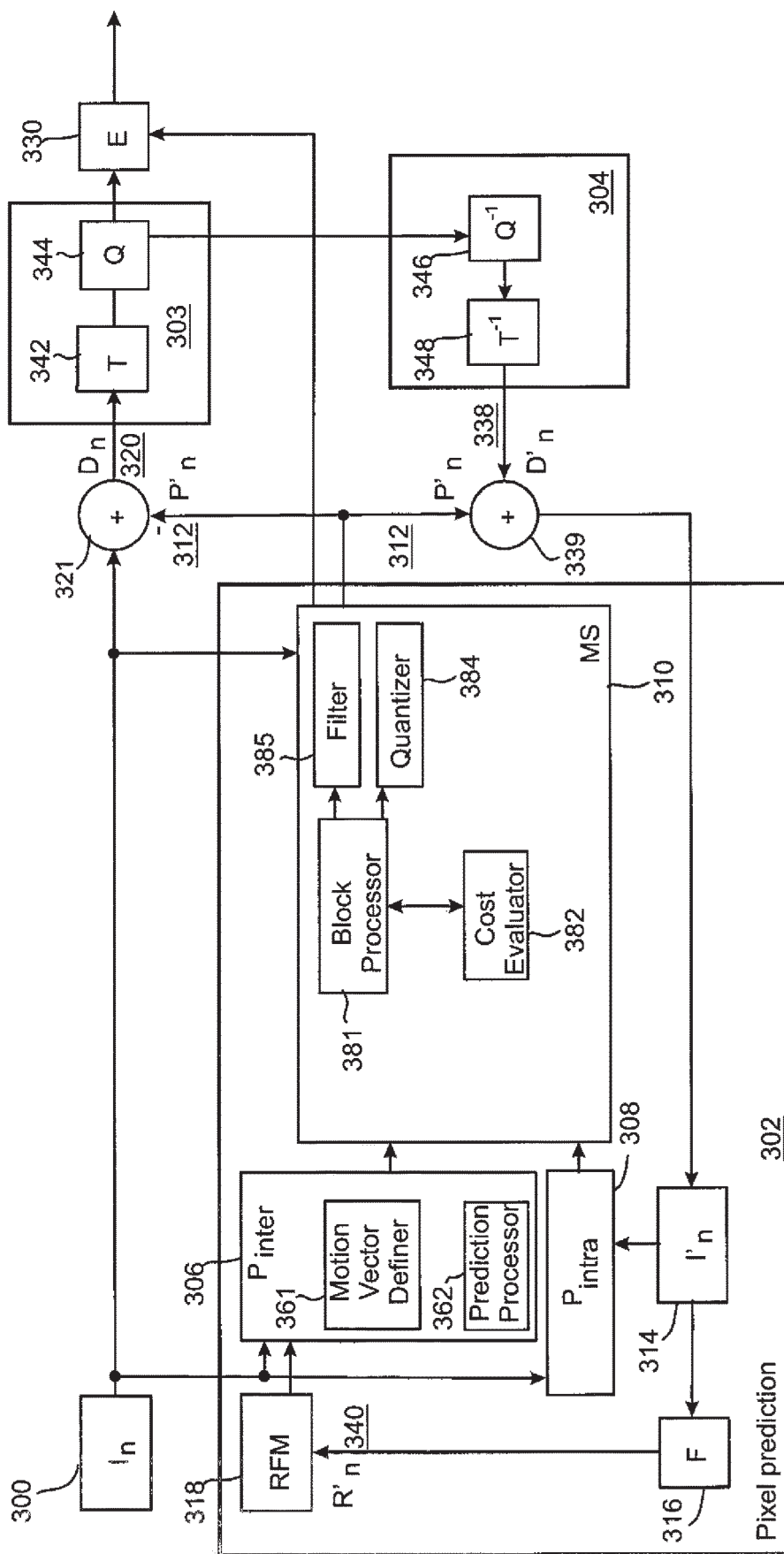


Fig. 4a

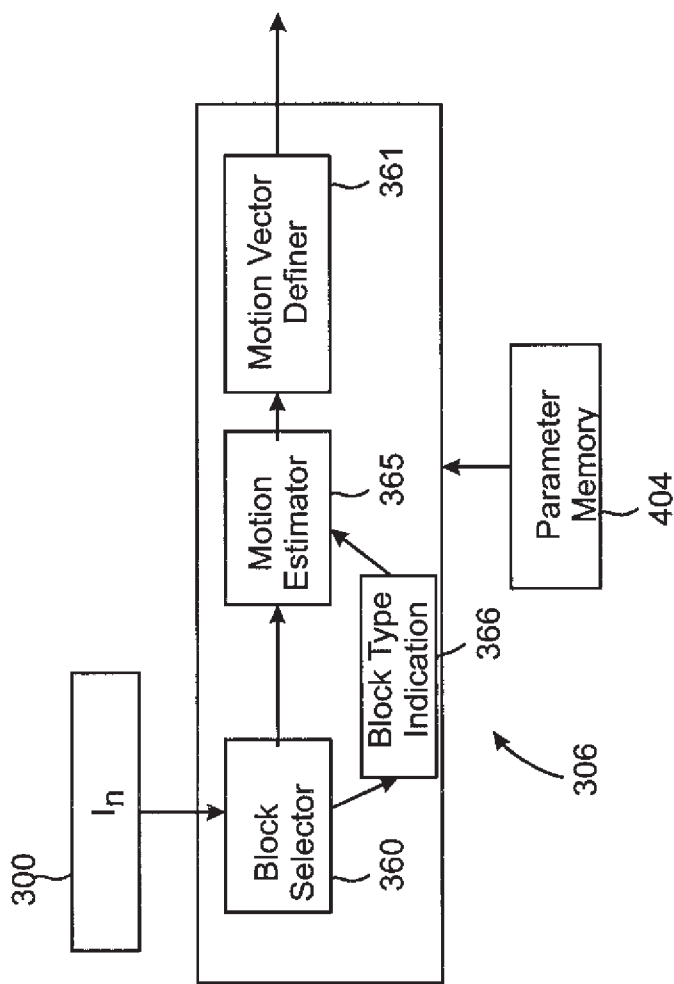


Fig. 4b

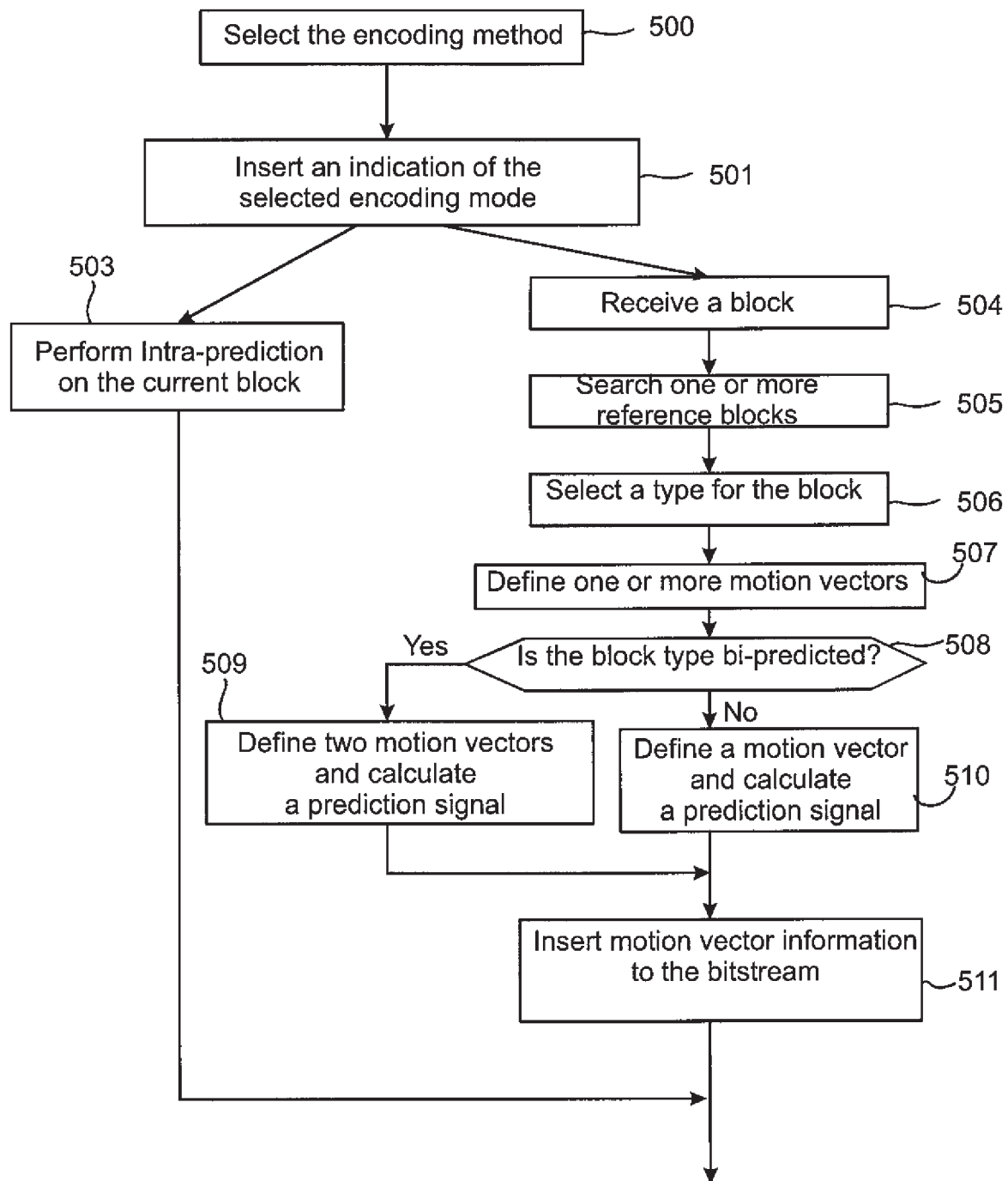


Fig. 5

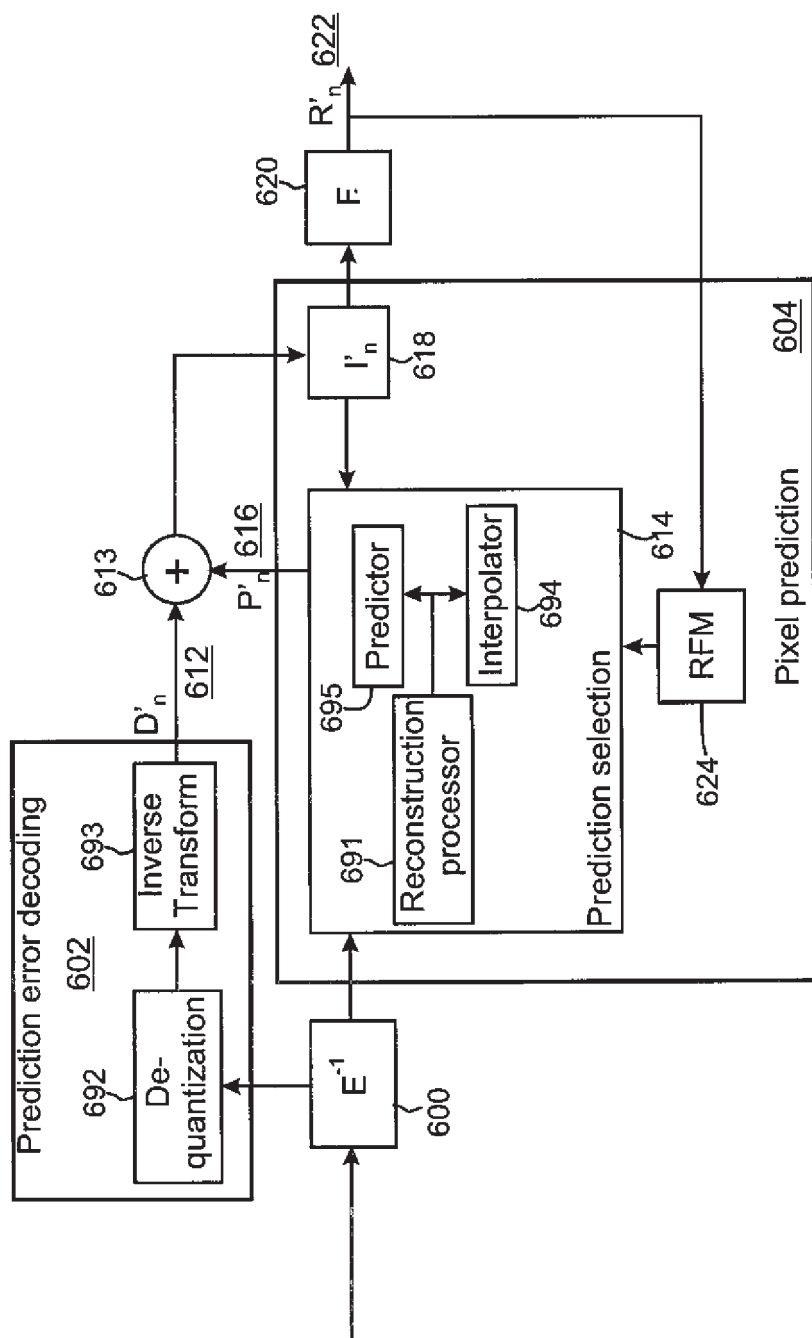


Fig. 6

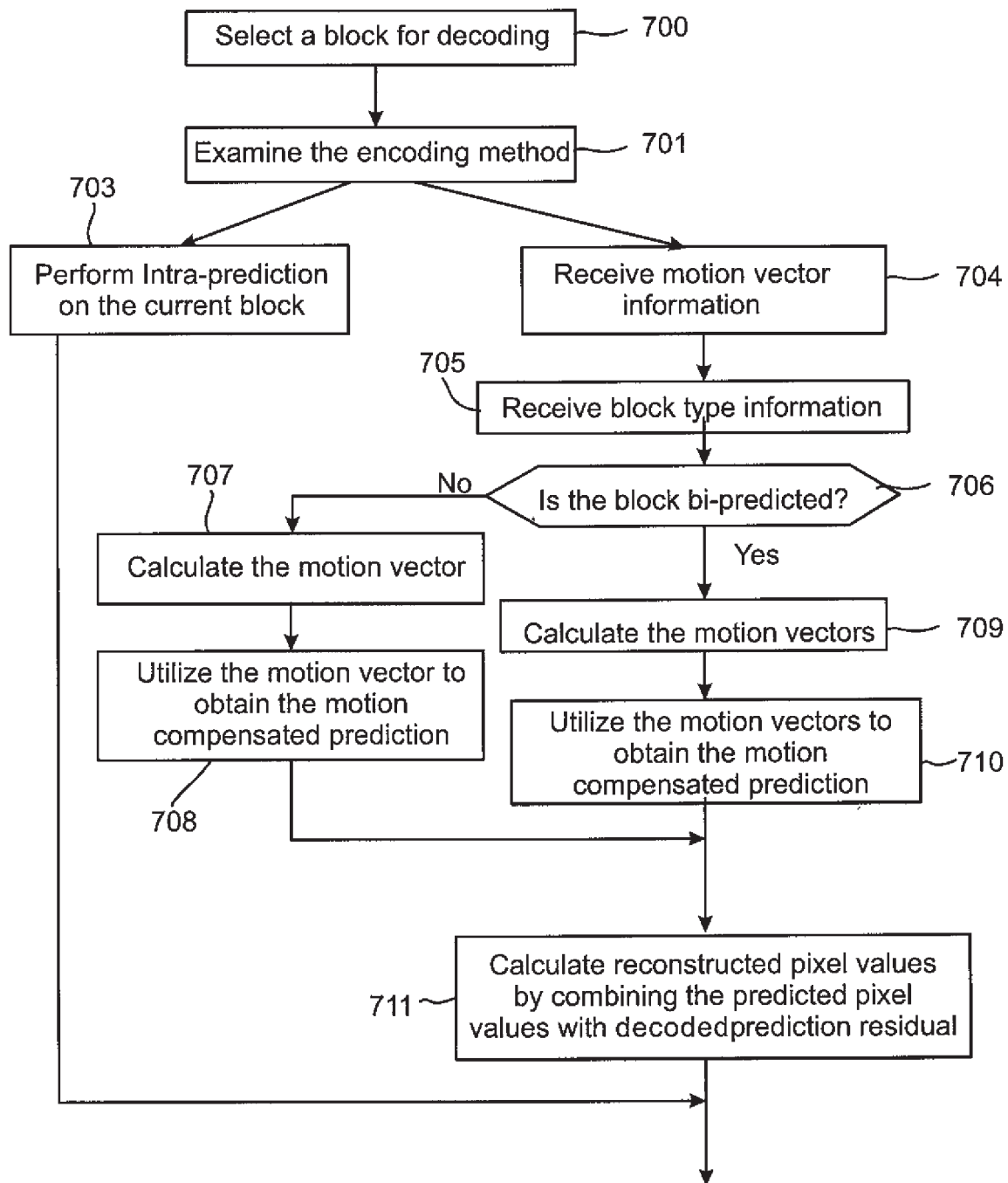


Fig. 7

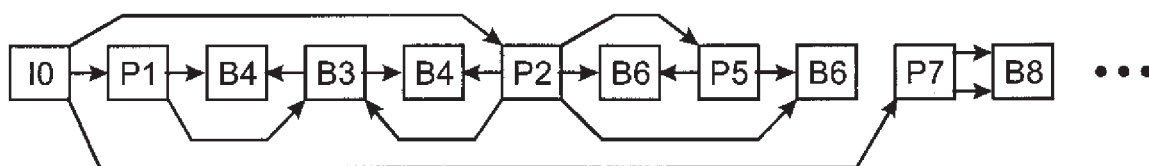


Fig. 8

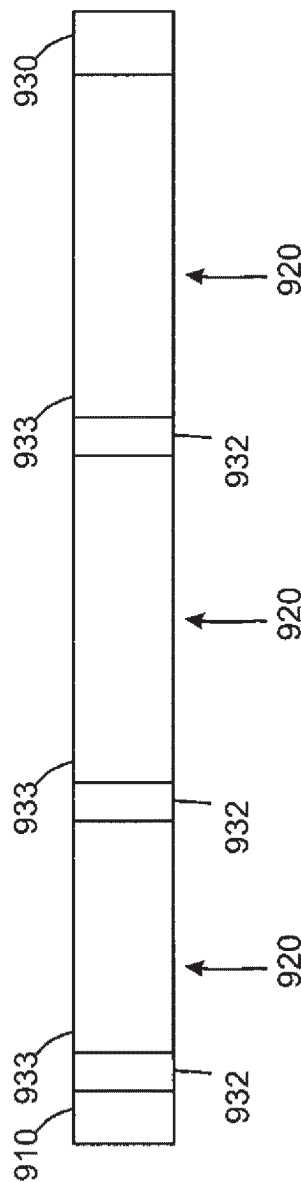


Fig. 9

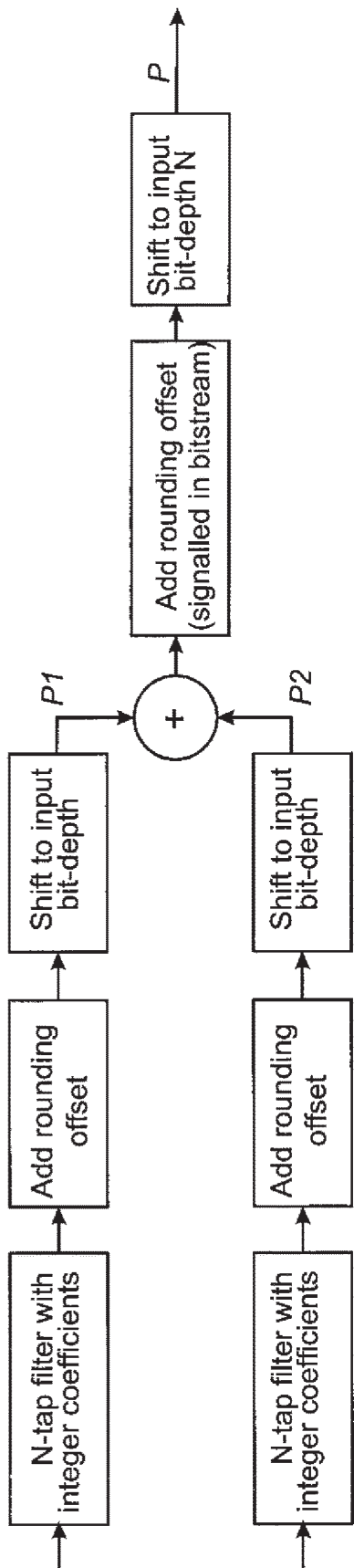


Fig. 10

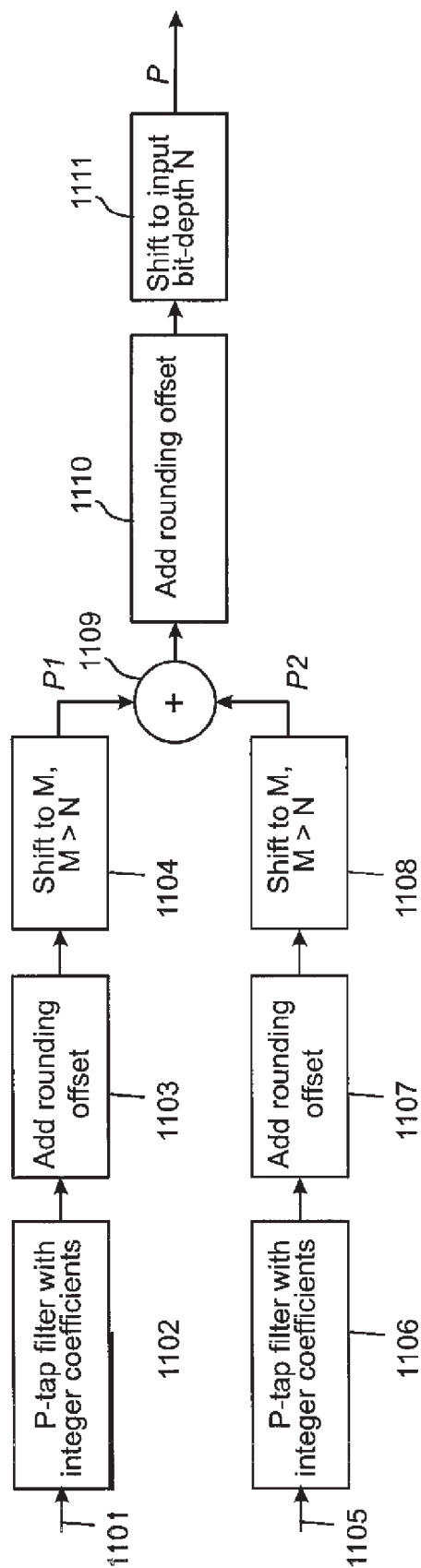


Fig. 11

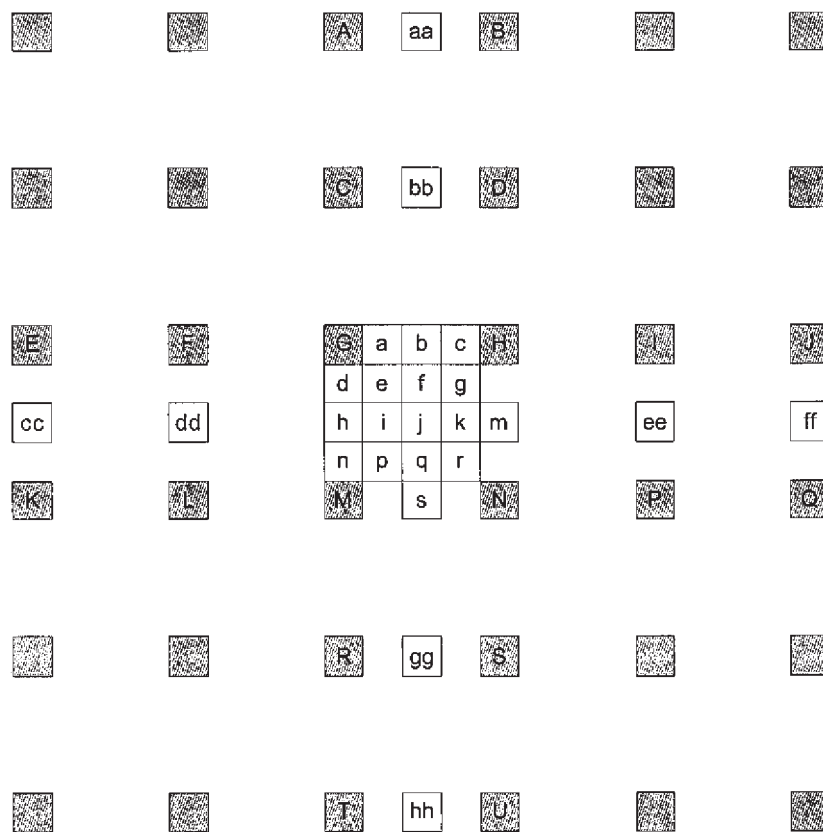


Fig. 12

1

MOTION PREDICTION IN VIDEO CODING**CROSS REFERENCE TO RELATED APPLICATIONS**

This application is a continuation of U.S. application Ser. No. 16/729,974, filed Dec. 30, 2019, which is a continuation of U.S. application Ser. No. 15/876,495, filed Jan. 22, 2018, which is a continuation of U.S. application Ser. No. 15/490,469, filed Apr. 18, 2017, which is a continuation of U.S. application Ser. No. 15/250,124, filed Aug. 29, 2016, which is a continuation of U.S. application Ser. No. 13/344,893, filed on Jan. 6, 2012, which claims priority to U.S. Provisional Application No. 61/430,694, filed Jan. 7, 2011, the entire contents of which are incorporated herein by reference.

TECHNICAL FIELD

The present invention relates to an apparatus, a method and a computer program for producing and utilizing motion prediction information in video encoding and decoding.

BACKGROUND INFORMATION

A video codec may comprise an encoder which transforms input video into a compressed representation suitable for storage and/or transmission and a decoder that can uncompress the compressed video representation back into a viewable form, or either one of them. The encoder may discard some information in the original video sequence in order to represent the video in a more compact form, for example at a lower bit rate.

Many hybrid video codecs, operating for example according to the International Telecommunication Union's ITU-T H.263 and H.264 coding standards, encode video information in two phases. In the first phase, pixel values in a certain picture area or "block" are predicted. These pixel values can be predicted, for example, by motion compensation mechanisms, which involve finding and indicating an area in one of the previously encoded video frames (or a later coded video frame) that corresponds closely to the block being coded. Additionally, pixel values can be predicted by spatial mechanisms which involve finding and indicating a spatial region relationship, for example by using pixel values around the block to be coded in a specified manner.

Prediction approaches using image information from a previous (or a later) image can also be called as Inter prediction methods, and prediction approaches using image information within the same image can also be called as Intra prediction methods.

The second phase is one of coding the error between the predicted block of pixels and the original block of pixels. This is typically accomplished by transforming the difference in pixel values using a specified transform. This transform may be e.g. a Discrete Cosine Transform (DCT) or a variant thereof. After transforming the difference, the transformed difference may be quantized and entropy encoded.

By varying the fidelity of the quantization process, the encoder can control the balance between the accuracy of the pixel representation, (in other words, the quality of the picture) and the size of the resulting encoded video representation (in other words, the file size or transmission bit rate).

An example of the encoding process is illustrated in FIG.

1.

2

The decoder reconstructs the output video by applying a prediction mechanism similar to that used by the encoder in order to form a predicted representation of the pixel blocks (using the motion or spatial information created by the encoder and stored in the compressed representation of the image) and prediction error decoding (the inverse operation of the prediction error coding to recover the quantized prediction error signal in the spatial domain).

After applying pixel prediction and error decoding processes the decoder combines the prediction and the prediction error signals (the pixel values) to form the output video frame.

The decoder (and encoder) may also apply additional filtering processes in order to improve the quality of the output video before passing it for display and/or storing as a prediction reference for the forthcoming frames in the video sequence.

An example of the decoding process is illustrated in FIG.

2.

Motion Compensated Prediction (MCP) is a technique used by video compression standards to reduce the size of an encoded bitstream. In MCP, a prediction for a current frame is formed using a previously coded frame(s), where only the difference between original and prediction signals, representative of the current and predicted frames, is encoded and sent to a decoder. A prediction signal, representative of a prediction frame, is formed by first dividing a current frame into blocks, e.g., macroblocks, and searching for a best match in a reference frame for each block. In this way, the motion of a block relative to the reference frame is determined and this motion information is coded into a bitstream as motion vectors. A decoder is able to reconstruct the exact prediction frame by decoding the motion vector data encoded in the bitstream.

An example of a prediction structure is presented in FIG. 8. Boxes indicate pictures, capital letters within boxes indicate coding types, numbers within boxes are picture numbers (in decoding order), and arrows indicate prediction dependencies. In this example I-pictures are intra pictures which do not use any reference pictures and thus can be decoded irrespective of the decoding of other pictures. P-pictures are so called uni-predicted pictures i.e. they refer to one reference picture, and B-pictures are bi-predicted pictures which use two other pictures as reference pictures, or two prediction blocks within one reference picture. In other words, the reference blocks relating to the B-picture may be in the same reference picture (as illustrated with the two arrows from picture P7 to picture B8 in FIG. 8) or in two different reference pictures (as illustrated e.g. with the arrows from picture P2 and from picture B3 to picture B4 in FIG. 8).

It should also be noted here that one picture may include different types of blocks i.e. blocks of a picture may be intra-blocks, uni-predicted blocks, and/or bi-predicted blocks. Motion vectors often relate to blocks wherein for one picture a plurality of motion vectors may exist.

In some systems the uni-predicted pictures are also called as uni-directionally predicted pictures and the bi-predicted pictures are called as bi-directionally predicted pictures.

The motion vectors are not limited to having full-pixel accuracy, but could have fractional-pixel accuracy as well. That is, motion vectors can point to fractional-pixel positions/locations of the reference frame, where the fractional-pixel locations can refer to, for example, locations "in between" image pixels. In order to obtain samples at fractional-pixel locations, interpolation filters may be used in the MCP process. Conventional video coding standards describe

how a decoder can obtain samples at fractional-pixel accuracy by defining an interpolation filter. In MPEG-2, for example, motion vectors can have at most, half-pixel accuracy, where the samples at half-pixel locations are obtained by a simple averaging of neighboring samples at full-pixel locations. The H.264/AVC video coding standard supports motion vectors with up to quarter-pixel accuracy. Furthermore, in the H.264/AVC video coding standard, half-pixel samples are obtained through the use of symmetric and separable 6-tap filters, while quarter-pixel samples are obtained by averaging the nearest half or full-pixel samples.

In typical video codecs, the motion information is indicated by motion vectors associated with each motion compensated image block. Each of these motion vectors represents the displacement of the image block in the picture to be coded (in the encoder) or decoded (at the decoder) and the prediction source block in one of the previously coded or decoded images (or pictures). In order to represent motion vectors efficiently, motion vectors are typically coded differentially with respect to block specific predicted motion vector. In a typical video codec, the predicted motion vectors are created in a predefined way, for example by calculating the median of the encoded or decoded motion vectors of the adjacent blocks.

In typical video codecs the prediction residual after motion compensation is first transformed with a transform kernel (like DCT) and then coded. The reason for this is that often there still exists some correlation among the residual and transform can in many cases help reduce this correlation and provide more efficient coding.

Typical video encoders utilize the Lagrangian cost function to find optimal coding modes, for example the desired macro block mode and associated motion vectors. This type of cost function uses a weighting factor or λ to tie together the exact or estimated image distortion due to lossy coding methods and the exact or estimated amount of information required to represent the pixel values in an image area.

This may be represented by the equation:

$$C = D + \lambda R \quad (1)$$

where C is the Lagrangian cost to be minimised, D is the image distortion (for example, the mean-squared error between the pixel values in original image block and in coded image block) with the mode and motion vectors currently considered, λ is a Lagrangian coefficient and R is the number of bits needed to represent the required data to reconstruct the image block in the decoder (including the amount of data to represent the candidate motion vectors).

Some hybrid video codecs, such as H.264/AVC, utilize bi-directional motion compensated prediction to improve the coding efficiency. In bi-directional prediction, prediction signal of the block may be formed by combining, for example by averaging two motion compensated prediction blocks. This averaging operation may further include either up or down rounding, which may introduce rounding errors.

The accumulation of rounding errors in bi-directional prediction may cause degradation in coding efficiency. This rounding error accumulation may be removed or decreased by signalling whether rounding up or rounding down have been used when the two prediction signals have been combined for each frame. Alternatively the rounding error could be controlled by alternating the usage of the rounding up and rounding down for each frame. For example, rounding up may be used for every other frame and, correspondingly, rounding down may be used for every other frame.

In FIG. 9 an example of averaging two motion compensated prediction blocks using rounding is illustrated. Sample

values of the first prediction reference is input **902** to a first filter **904** in which values of two or more full pixels near the point which the motion vector is referring to are used in the filtering. A rounding offset may be added **906** to the filtered value. The filtered value added with the rounding offset is right shifted **908** x-bits i.e. divided by 2^x to obtain a first prediction signal **P1**. Similar operation is performed to the second prediction reference as is illustrated with blocks **912**, **914**, **916** and **918** to obtain a second prediction signal **P2**. The first prediction signal **P1** and the second prediction signal **P2** are combined e.g. by summing the prediction signals **P1**, **P2**. A rounding offset may be added **920** with the combined signal after which the result is right shifted y-bits i.e. divided by 2^y . The rounding may be upwards, if the rounding offset is positive, or downwards, if the rounding offset is negative. The direction of the rounding may always be the same, or it may alter from time to time, e.g. for each frame. The direction of the rounding may be signaled in the bitstream so that in the decoding process the same rounding direction can be used.

However, these methods increase somewhat the complexity as two separate code branches need to be written for bi-directional averaging. In addition, the motion estimation routines in the encoder may need to be doubled for both cases of rounding and truncation.

SUMMARY

The present invention introduces a method which enables reducing the effect of rounding errors in bi-directional and multi-directional prediction. According to some embodiments of the invention prediction signals are maintained in a higher precision during the prediction calculation and the precision is reduced after the two or more prediction signals have been combined with each other.

In some example embodiments prediction signals are maintained in higher accuracy until the prediction signals have been combined to obtain the bi-directional or multidirectional prediction signal. The accuracy of the bi-directional or multidirectional prediction signal can then be downshifted to an appropriate accuracy for post processing purposes. Then, no rounding direction indicator need not be included in or read from the bitstream.

According to a first aspect of the present invention there is provided a method comprising:

determining a block of pixels of a video representation encoded in a bitstream, values of said pixels having a first precision;

determining a type of the block;

if the determining indicates that the block is a block predicted by using two or more reference blocks,

determining a first reference pixel location in a first reference block and a second reference pixel location in a second reference block;

using said first reference pixel location to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

using said second reference pixel location to obtain a second prediction, said second prediction having the second precision, which is higher than said first precision;

combining said first prediction and said second prediction to obtain a combined prediction; and

decreasing the precision of said combined prediction to said first precision.

According to a second aspect of the present invention there is provided an apparatus comprising:

a processor; and

5

a memory unit operatively connected to the processor and including:

computer code configured to determine a block of pixels of a video representation encoded in a bitstream, values of said pixels having a first precision;

computer code configured to determine a type of the block;

computer code configured to, if the determining indicates that the block is a block predicted by using two or more reference blocks,

determine a first reference pixel location in a first reference block and a second reference pixel location in a second reference block;

use said first reference pixel location to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

use said second reference pixel location to obtain a second prediction, said second prediction having the second precision, which is higher than said first precision;

combine said first prediction and said second prediction to obtain a combined prediction; and

decrease the precision of said combined prediction to said first precision.

According to a third aspect of the present invention there is provided a computer readable storage medium stored with code thereon for use by an apparatus, which when executed by a processor, causes the apparatus to perform:

determine a block of pixels of a video representation encoded in a bitstream, values of said pixels having a first precision;

determine a type of the block;

if the determining indicates that the block is a block predicted by using two or more reference blocks,

determine a first reference pixel location in a first reference block and a second reference pixel location in a second reference block;

use said first reference pixel location to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

use said second reference pixel location to obtain a second prediction, said second prediction having the second precision, which is higher than said first precision;

combine said first prediction and said second prediction to obtain a combined prediction; and

decrease the precision of said combined prediction to said first precision.

According to a fourth aspect of the present invention there is provided at least one processor and at least one memory, said at least one memory stored with code thereon, which when executed by said at least one processor, causes an apparatus to perform:

determine a block of pixels of a video representation encoded in a bitstream, values of said pixels having a first precision;

determine a type of the block;

if the determining indicates that the block is a block predicted by using two or more reference blocks,

determine a first reference pixel location in a first reference block and a second reference pixel location in a second reference block;

use said first reference pixel location to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

use said second reference pixel location to obtain a second prediction, said second prediction having the second precision, which is higher than said first precision;

6

combine said first prediction and said second prediction to obtain a combined prediction; and

decrease the precision of said combined prediction to said first precision.

According to a fifth aspect of the present invention there is provided an apparatus comprising:

an input to determine a block of pixels of a video representation encoded in a bitstream, values of said pixels having a first precision;

a determinator to determine a type of the block; wherein if the determining indicates that the block is a block predicted by using two or more reference blocks, said determinator further to determine a first reference pixel location in a first reference block and a second reference pixel location in a second reference block;

a first predictor to use said first reference pixel location to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

a second predictor to use said second reference pixel location to obtain a second prediction, said second prediction having the second precision, which is higher than said first precision;

a combiner to combine said first prediction and said second prediction to obtain a combined prediction; and

a shifter to decrease the precision of said combined prediction to said first precision.

According to a sixth aspect of the present invention there is provided an apparatus comprising:

means for determining a block of pixels of a video representation encoded in a bitstream, values of said pixels having a first precision;

means for determining a type of the block;

means for determining a first reference pixel location in a first reference block and a second reference pixel location in a second reference block, if the determining indicates that the block is a block predicted by using two or more reference blocks;

means for using said first reference pixel location to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

means for using said second reference pixel location to obtain a second prediction, said second prediction having the second precision, which is higher than said first precision;

means for combining said first prediction and said second prediction to obtain a combined prediction; and

means for decreasing the precision of said combined prediction to said first precision.

This invention removes the need to signal the rounding offset or use different methods for rounding for different frames. This invention may keep the motion compensated prediction signal of each one of the predictions at highest precision possible after interpolation and perform the rounding to the bit-depth range of the video signal after both prediction signals are added.

DESCRIPTION OF THE DRAWINGS

For better understanding of the present invention, reference will now be made by way of example to the accompanying drawings in which:

FIG. 1 shows schematically an electronic device employing some embodiments of the invention;

FIG. 2 shows schematically a user equipment suitable for employing some embodiments of the invention;

7

FIG. 3 further shows schematically electronic devices employing embodiments of the invention connected using wireless and wired network connections;

FIG. 4a shows schematically an embodiment of the invention as incorporated within an encoder;

FIG. 4b shows schematically an embodiment of an inter predictor according to some embodiments of the invention;

FIG. 5 shows a flow diagram showing the operation of an embodiment of the invention with respect to the encoder as shown in FIG. 4a;

FIG. 6 shows a schematic diagram of a decoder according to some embodiments of the invention;

FIG. 7 shows a flow diagram of showing the operation of an embodiment of the invention with respect to the decoder shown in FIG. 6;

FIG. 8 illustrates an example of a prediction structure in a video sequence;

FIG. 9 depicts an example of a bit stream of an image;

FIG. 10 depicts an example of bi-directional prediction using rounding;

FIG. 11 depicts an example of bi-directional prediction according to an example embodiment of the present invention; and

FIG. 12 illustrates an example of some possible prediction directions for a motion vector.

DETAILED DESCRIPTION OF SOME EXAMPLE EMBODIMENTS

The following describes in further detail suitable apparatus and possible mechanisms for the provision of reducing information to be transmitted in video coding systems and more optimal codeword mappings in some embodiments. In this regard reference is first made to FIG. 1 which shows a schematic block diagram of an exemplary apparatus or electronic device 50, which may incorporate a codec according to an embodiment of the invention.

The electronic device 50 may for example be a mobile terminal or user equipment of a wireless communication system. However, it would be appreciated that embodiments of the invention may be implemented within any electronic device or apparatus which may require encoding and decoding or encoding or decoding video images.

The apparatus 50 may comprise a housing 30 for incorporating and protecting the device. The apparatus 50 further may comprise a display 32 in the form of a liquid crystal display. In other embodiments of the invention the display may be any suitable display technology suitable to display an image or video. The apparatus 50 may further comprise a keypad 34. In other embodiments of the invention any suitable data or user interface mechanism may be employed. For example the user interface may be implemented as a virtual keyboard or data entry system as part of a touch-sensitive display. The apparatus may comprise a microphone 36 or any suitable audio input which may be a digital or analogue signal input. The apparatus 50 may further comprise an audio output device which in embodiments of the invention may be any one of: an earpiece 38, speaker, or an analogue audio or digital audio output connection. The apparatus 50 may also comprise a battery 40 (or in other embodiments of the invention the device may be powered by any suitable mobile energy device such as solar cell, fuel cell or clockwork generator). The apparatus may further comprise an infrared port 42 for short range line of sight communication to other devices. In other embodiments the apparatus 50 may further comprise any suitable short range

8

communication solution such as for example a Bluetooth wireless connection or a USB/firewire wired connection.

The apparatus 50 may comprise a controller 56 or processor for controlling the apparatus 50. The controller 56 may be connected to memory 58 which in embodiments of the invention may store both data in the form of image and audio data and/or may also store instructions for implementation on the controller 56. The controller 56 may further be connected to codec circuitry 54 suitable for carrying out coding and decoding of audio and/or video data or assisting in coding and decoding carried out by the controller 56.

The apparatus 50 may further comprise a card reader 48 and a smart card 46, for example a UICC and UICC reader for providing user information and being suitable for providing authentication information for authentication and authorization of the user at a network.

The apparatus 50 may comprise radio interface circuitry 52 connected to the controller and suitable for generating wireless communication signals for example for communication with a cellular communications network, a wireless communications system or a wireless local area network. The apparatus 50 may further comprise an antenna 44 connected to the radio interface circuitry 52 for transmitting radio frequency signals generated at the radio interface circuitry 52 to other apparatus(es) and for receiving radio frequency signals from other apparatus(es).

In some embodiments of the invention, the apparatus 50 comprises a camera capable of recording or detecting individual frames which are then passed to the codec 54 or controller for processing. In some embodiments of the invention, the apparatus may receive the video image data for processing from another device prior to transmission and/or storage. In some embodiments of the invention, the apparatus 50 may receive either wirelessly or by a wired connection the image for coding/decoding.

With respect to FIG. 3, an example of a system within which embodiments of the present invention can be utilized is shown. The system 10 comprises multiple communication devices which can communicate through one or more networks. The system 10 may comprise any combination of wired or wireless networks including, but not limited to a wireless cellular telephone network (such as a GSM, UMTS, CDMA network etc), a wireless local area network (WLAN) such as defined by any of the IEEE 802.x standards, a Bluetooth personal area network, an Ethernet local area network, a token ring local area network, a wide area network, and the Internet.

The system 10 may include both wired and wireless communication devices or apparatus 50 suitable for implementing embodiments of the invention.

For example, the system shown in FIG. 3 shows a mobile telephone network 11 and a representation of the internet 28. Connectivity to the internet 28 may include, but is not limited to, long range wireless connections, short range wireless connections, and various wired connections including, but not limited to, telephone lines, cable lines, power lines, and similar communication pathways.

The example communication devices shown in the system 10 may include, but are not limited to, an electronic device or apparatus 50, a combination of a personal digital assistant (PDA) and a mobile telephone 14, a PDA 16, an integrated messaging device (IMD) 18, a desktop computer 20, a notebook computer 22. The apparatus 50 may be stationary or mobile when carried by an individual who is moving. The apparatus 50 may also be located in a mode of transport including, but not limited to, a car, a truck, a taxi, a bus, a

train, a boat, an airplane, a bicycle, a motorcycle or any similar suitable mode of transport.

Some or further apparatus may send and receive calls and messages and communicate with service providers through a wireless connection **25** to a base station **24**. The base station **24** may be connected to a network server **26** that allows communication between the mobile telephone network **11** and the internet **28**. The system may include additional communication devices and communication devices of various types.

The communication devices may communicate using various transmission technologies including, but not limited to, code division multiple access (CDMA), global systems for mobile communications (GSM), universal mobile telecommunications system (UMTS), time divisional multiple access (TDMA), frequency division multiple access (FDMA), transmission control protocol-internet protocol (TCP-IP), short messaging service (SMS), multimedia messaging service (MMS), email, instant messaging service (IMS), Bluetooth, IEEE 802.11 and any similar wireless communication technology. A communications device involved in implementing various embodiments of the present invention may communicate using various media including, but not limited to, radio, infrared, laser, cable connections, and any suitable connection.

Various embodiments can extend conventional two-stage sub-pixel interpolation algorithms, such as the algorithm used in the H.264/AVC video coding standard, without the need to increase the complexity of the decoder. It should be noted here that FIG. **11** illustrates only some full pixel values which are the nearest neighbors to the example block of pixels but in the interpolation it may also be possible to use full pixel values located farther from the block under consideration. Furthermore, the present invention is not only limited to implementations using one-dimensional interpolation but the fractional pixel samples can also be obtained using more complex interpolation or filtering.

It should be noted that various embodiments can be implemented by and/or in conjunction with other video coding standards besides the H.264/AVC video coding standard.

With respect to FIG. **4a**, a block diagram of a video encoder suitable for carrying out embodiments of the invention is shown. Furthermore, with respect to FIG. **5**, the operation of the encoder exemplifying embodiments of the invention specifically with respect to the utilization of higher accuracy calculation of prediction signals is shown as a flow diagram.

FIG. **4a** shows the encoder as comprising a pixel predictor **302**, prediction error encoder **303** and prediction error decoder **304**. FIG. **4a** also shows an embodiment of the pixel predictor **302** as comprising an inter-predictor **306**, an intra-predictor **308**, a mode selector **310**, a filter **316**, and a reference frame memory **318**. The mode selector **310** comprises a block processor **381** and a cost evaluator **382**. FIG. **4b** also depicts an embodiment of the inter-predictor **306** which comprises a block selector **360** and a motion vector definer **361**, which may be implemented e.g. in a prediction processor **362**. The inter-predictor **306** may also have access to a parameter memory **404**. The mode selector **310** may also comprise a quantizer **384**.

The pixel predictor **302** receives the image **300** to be encoded at both the inter-predictor **306** (which determines the difference between the image and a motion compensated reference frame **318**) and the intra-predictor **308** (which determines a prediction for an image block based only on the already processed parts of current frame or picture). The

output of both the inter-predictor and the intra-predictor are passed to the mode selector **310**. The intra-predictor **308** may have more than one intra-prediction modes. Hence, each mode may perform the intra-prediction and provide the predicted signal to the mode selector **310**. The mode selector **310** also receives a copy of the image **300**.

The block processor **381** determines which encoding mode to use to encode the current block. If the block processor **381** decides to use an inter-prediction mode it will pass the output of the inter-predictor **306** to the output of the mode selector **310**. If the block processor **381** decides to use an intra-prediction mode it will pass the output of one of the intra-predictor modes to the output of the mode selector **310**.

According to some example embodiments the pixel predictor **302** operates as follows. The inter predictor **306** and the intra prediction modes **308** perform the prediction of the current block to obtain predicted pixel values of the current block. The inter predictor **306** and the intra prediction modes **308** may provide the predicted pixel values of the current block to the block processor **381** for analyzing which prediction to select. In addition to the predicted values of the current block, the block processor **381** may, in some embodiments, receive an indication of a directional intra prediction mode from the intra prediction modes.

The block processor **381** examines whether to select the inter prediction mode or the intra prediction mode. The block processor **381** may use cost functions such as the equation (1) or some other methods to analyze which encoding method gives the most efficient result with respect to a certain criterion or criteria. The selected criteria may include coding efficiency, processing costs and/or some other criteria. The block processor **381** may examine the prediction for each directionality i.e. for each intra prediction mode and inter prediction mode and calculate the cost value for each intra prediction mode and inter prediction mode, or the block processor **381** may examine only a subset of all available prediction modes in the selection of the prediction mode.

In some embodiments the inter predictor **306** operates as follows. The block selector **360** receives a current block to be encoded (block **504** in FIG. **5**) and examines whether a previously encoded image contains a block which may be used as a reference to the current block (block **505**). If such a block is found from the reference frame memory **318**, the motion estimator **365** may determine whether the current block could be predicted by using one or two (or more) reference blocks i.e. whether the current block could be a uni-predicted block or a bi-predicted block (block **506**). If the motion estimator **365** has determined to use uni-prediction, the motion estimator **365** may indicate the reference block to the motion vector definer **361**. If the motion estimator **365** has selected to use bi-prediction, the motion estimator **365** may indicate both reference blocks, or if more than two reference blocks have been selected, all the selected reference blocks to the motion vector definer **361**. The motion vector definer **361** utilizes the reference block information and defines a motion vector (block **507**) to indicate the correspondence between pixels of the current block and the reference block(s).

In some embodiments the inter predictor **306** calculates a cost value for both one-directional and bi-directional prediction and may then select which kind of prediction to use with the current block.

In some embodiments the motion vector may point to a full pixel sample or to a fraction pixel sample i.e. to a half pixel, to a quarter pixel or to a one-eighth pixel. The motion vector definer **361** may examine the type of the current block

11

to determine whether the block is a bi-predicted block or another kind of a block (block **508**). The type may be determined by the block type indication **366** which may be provided by the block selector **360** or another element of the encoder. If the type of the block is a bi-predicted block, two (or more) motion vectors are defined by the motion vector definer **361** (block **509**). Otherwise, if the block is a uni-predicted block, one motion vector shall be defined (block **510**).

It is also possible that the type of the block is determined before the motion vector is calculated.

The motion vector definer **361** provides motion vector information to the block processor **381** which uses this information to obtain the prediction signal.

When the cost has been calculated with respect to intra prediction mode and possibly with respect to the inter prediction mode(s), the block processor **381** selects one intra prediction mode or the inter prediction mode for encoding the current block.

When the inter prediction mode was selected, the predicted pixel values or predicted pixel values quantized by the optional quantizer **384** are provided as the output of the mode selector.

The output of the mode selector is passed to a first summing device **321**. The first summing device may subtract the pixel predictor **302** output from the image **300** to produce a first prediction error signal **320** which is input to the prediction error encoder **303**.

The pixel predictor **302** further receives from a preliminary reconstructor **339** the combination of the prediction representation of the image block **312** and the output **338** of the prediction error decoder **304**. The preliminary reconstructed image **314** may be passed to the intra-predictor **308** and to a filter **316**. The filter **316** receiving the preliminary representation may filter the preliminary representation and output a final reconstructed image **340** which may be saved in a reference frame memory **318**. The reference frame memory **318** may be connected to the inter-predictor **306** to be used as the reference image against which the future image **300** is compared in inter-prediction operations.

The operation of the pixel predictor **302** may be configured to carry out any known pixel prediction algorithm known in the art.

The pixel predictor **302** may also comprise a filter **385** to filter the predicted values before outputting them from the pixel predictor **302**.

The operation of the prediction error encoder **303** and prediction error decoder **304** will be described hereafter in further detail. In the following examples the encoder generates images in terms of 16×16 pixel macroblocks which go to form the full image or picture. Thus, for the following examples the pixel predictor **302** outputs a series of predicted macroblocks of size 16×16 pixels and the first summing device **321** outputs a series of 16×16 pixel residual data macroblocks which may represent the difference between a first macro-block in the image **300** against a predicted macro-block (output of pixel predictor **302**). It would be appreciated that other size macro blocks may be used.

The prediction error encoder **303** comprises a transform block **342** and a quantizer **344**. The transform block **342** transforms the first prediction error signal **320** to a transform domain. The transform is, for example, the DCT transform. The quantizer **344** quantizes the transform domain signal, e.g. the DCT coefficients, to form quantized coefficients.

The entropy encoder **330** receives the output of the prediction error encoder and may perform a suitable entropy

12

encoding/variable length encoding on the signal to provide error detection and correction capability. Any suitable entropy encoding algorithm may be employed.

The prediction error decoder **304** receives the output from the prediction error encoder **303** and performs the opposite processes of the prediction error encoder **303** to produce a decoded prediction error signal **338** which when combined with the prediction representation of the image block **312** at the second summing device **339** produces the preliminary reconstructed image **314**. The prediction error decoder may be considered to comprise a dequantizer **346**, which dequantizes the quantized coefficient values, e.g. DCT coefficients, to reconstruct the transform signal and an inverse transformation block **348**, which performs the inverse transformation to the reconstructed transform signal wherein the output of the inverse transformation block **348** contains reconstructed block(s). The prediction error decoder may also comprise a macroblock filter (not shown) which may filter the reconstructed macroblock according to further decoded information and filter parameters.

The operation and implementation of the mode selector **310** is shown in further detail with respect to FIG. 5. On the basis of the prediction signals from the output of the inter-predictor **306**, the output of the intra-predictor **308** and/or the image signal **300** the block processor **381** determines which encoding mode to use to encode the current image block. This selection is depicted as the block **500** in FIG. 5. The block processor **381** may calculate a rate-distortion cost (RD) value or another cost value for the prediction signals which are input to the mode selector **310** and select such an encoding mode **503**, **504** for which the determined cost is the smallest.

The mode selector **310** provides an indication of the encoding mode of the current block (**501**). The indication may be encoded and inserted to a bit stream or stored into a memory together with the image information.

If the intra-prediction mode is selected, the block is predicted by an intra-prediction method (**503**). Respectively, if the inter-prediction mode is selected, the block is predicted by an inter-prediction method (**504-510**).

An example of the operation of the mode selector when the inter-prediction mode is selected and the type of the block is a bi-predicted block, is illustrated as a block diagram in FIG. 11. Motion vector information provided by the motion vector definer **361** contains indication of a first reference block and a second reference block. In multi-prediction applications the motion vector information may contain indication of more than two reference blocks. The block processor **381** uses the motion vector information to determine which block is used as a first reference block for the current block and which block is used as a second reference block for the current block. The block processor **381** then uses some pixel values of the first reference block to obtain first prediction values and some pixel values of the second reference block to obtain second prediction values. For example, if a first motion vector points to a fraction of a pixel (a subpixel) illustrated by the square b in the example of FIG. 12, the block processor **381** may use pixel values of several full pixels on the same row, for example, than said fraction of the pixel to obtain a reference pixel value. The block processor **381** may use e.g. a P-tap filter such as a six-tap filter in which P pixel values of the reference block are used to calculate the prediction value. In the example of FIG. 12 these pixel values could be pixels E, F, G, H, I and J. The taps of the filter may be e.g. integer values. An example of such a six-tap filter is [1-5 20 20-5 1]/32. Hence, the filter **1102** would receive **1101** the pixel values of

13

pixels E, F, G, H, I and J and filter these values by the equation $P1=(E_1-5*F_1+20*G_1+20*H_1-5*I_1+J_1)$, in which E_1 is the value of the pixel E in the first reference block, F_1 is the value of the pixel F in the first reference block, G_1 is the value of the pixel G in the first reference block, H_1 is the value of the pixel H in the first reference block, I_1 is the value of the pixel I in the first reference block, and J_1 is the value of the pixel J in the first reference block. In the first rounding offset insertion block **1103** a first rounding offset may be added to the value $P1$ i.e. $P1+\text{rounding offset}$. Then, the sum may be shifted by the first shifting block **1104** to the right so that the precision of the sum becomes M bits. The precision M is higher than the precision of the expected prediction value. For example, pixel values and the prediction values may be represented by N bits wherein $M>N$. In some example implementations N is 8 bits and M is 16 bits but it is obvious that also other bit lengths can be used with the present invention.

The second prediction can be obtained similarly by the second filter **1106**, which receives **1105** some pixel values of the second reference block. These pixel values are determined on the basis of the second motion vector. The second motion vector may point to the same pixel (or a fraction of the pixel) in the second reference block to which the first motion vector points in the first reference block (using the example above that pixel is the subpixel b) or to another full pixel or a subpixel in the second reference block. The second filter **1106** uses similar filter than the first filter **1102** and outputs the second filtering result $P2$. According to the example above the filter is a six-tap filter $[1 \ -5 \ 20 \ 20 \ -5 \ 1]/32$, wherein $P2=(E_2-5*F_2+20*G_2+20*H_2-5*I_2+J_2)$, in which E_2 is the value of the pixel E in the second reference block, F_2 is the value of the pixel F in the second reference block, G_2 is the value of the pixel G in the second reference block, H_2 is the value of the pixel H in the second reference block, I_2 is the value of the pixel I in the second reference block, and J_2 is the value of the pixel J in the second reference block. In the second rounding offset insertion block **1107** the first rounding offset may be added to the value $P2$ i.e. $P2+\text{rounding offset}$. Then, the sum may be shifted by the second shifting block **1108** to the right so that the precision of the sum becomes M bits.

In the combining block **1109** the two prediction values $P1$, $P2$ are combined e.g. by summing and the combined value is added with a second rounding value in the third rounding value insertion block **1110**. The result is converted to a smaller precision e.g. by shifting bits of the result to the right y times in the third shifting block **1111**. This corresponds with dividing the result by 2^y . After the conversion the precision of the prediction signal corresponds with the precision of the input pixel values. However, the intermediate results are at a higher precision, wherein possible rounding errors have a smaller effect to the prediction signal compared to existing methods such as the method illustrated in FIG. 10.

In an alternative embodiment the rounding offset is not added separately to the results of the first **1102** and the second filter **1106** but after combining the results in the combining block **1110**. In this case the value of the rounding offset is twice the value of the first rounding offset because in the embodiment of FIG. 11 the first rounding offset is actually added twice, once to $P1$ and once to $P2$.

In some embodiments also the first shifting block **1105** and the second shifting block **1109** are not needed when the precision of registers which store the filtering results is sufficient without reducing the precision of the filtering results. In that case the third shifting block may need to shift

14

the prediction result more than y bits to the right so that the right shifted value P has the same prediction than the input pixel values, for example 8 bits.

In some other example embodiments may partly differ from the above. For example, if a motion vector of one of the prediction directions point to an integer sample, the bit-depth of prediction samples with integer accuracy may be increased by shifting the samples to the left so that the filtering can be performed with values having the same precision.

Samples of each one of the prediction directions could be rounded at an intermediate step to a bit-depth that is still larger than the input bit-depth to make sure all the intermediate values fit to registers of certain length, e.g. 16-bit registers. For example, let's consider the same example above but using filter taps: {3, -17, 78, 78, -17, 3}. Then $P1$ and $P2$ are obtained as:

$$P1=(3*E_1-17*F_1+78*G_1+78*H_1-17*I_1+3*J_1)>>1$$

$$P2=(3*E_2-17*F_2+78*G_2+78*H_2-17*I_2+3*J_2)>>1$$

The bi-directional prediction signal may then be obtained using:

$$P=(P1+P2+32)>>6.$$

When a motion vector points between two full pixels i.e. to a fraction of the pixel, the value for that the reference pixel value may be obtained in several ways. Some possibilities were disclosed above but in the following some further non-limiting examples shall be provided with reference to FIG. 12.

If a motion vector points to the block labeled j the corresponding reference pixel value could be obtained by using full pixel values on the same diagonal than j, or by a two-phase process in which e.g. pixel values of rows around the block j are used to calculate a set of intermediate results and then these intermediate results could be filtered to obtain the reference pixel value. In an example embodiment the full pixel values A and B could be used to calculate a first intermediate result to represent a fraction pixel value aa, full pixel values C and D could be used to calculate a second intermediate result to represent a fraction pixel value bb, and full pixel values E to J could be used to calculate a third intermediate result to represent a fraction pixel value b. Similarly, fourth, fifth and sixth intermediate values to represent fraction pixel values s, gg, hh could be calculated on the basis of full pixel values K to Q; R, S; and T, U. These intermediate results could then be filtered by a six-tap filter, for example.

The prediction signal P obtained by the above described operations need not be provided to a decoder but the encoder uses this information to obtain predicted blocks and prediction error. The prediction error may be provided to the decoder so that the decoder can use corresponding operations to obtain the predicted blocks by prediction and correct the prediction results on the basis of the prediction error. The encoder may also provide motion vector information to the decoder.

In an example embodiment, as is depicted in FIG. 9, the bit stream of an image comprises an indication of the beginning of an image **910**, image information of each block of the image **920**, and indication of the end of the image **930**. The image information of each block of the image **920** may include a block type indicator **932**, and motion vector information **933**. It is obvious that the bit stream may also comprise other information. Further, this is only a simplified

15

image of the bit stream and in practical implementations the contents of the bit stream may be different from what is depicted in FIG. 9.

The bit stream may further be encoded by the entropy encoder 330.

Although the embodiments above have been described with respect to the size of the macroblock being 16×16 pixels, it would be appreciated that the methods and apparatus described may be configured to handle macroblocks of different pixel sizes.

In the following the operation of an example embodiment of the decoder 600 is depicted in more detail with reference to FIG. 6.

At the decoder side similar operations are performed to reconstruct the image blocks. FIG. 6 shows a block diagram of a video decoder suitable for employing embodiments of the invention and FIG. 7 shows a flow diagram of an example of a method in the video decoder. The decoder shows an entropy decoder 600 which performs an entropy decoding on the received signal. The entropy decoder thus performs the inverse operation to the entropy encoder 330 of the encoder described above. The entropy decoder 600 outputs the results of the entropy decoding to a prediction error decoder 602 and a pixel predictor 604.

The pixel predictor 604 receives the output of the entropy decoder 600. The output of the entropy decoder 600 may include an indication on the prediction mode used in encoding the current block. A predictor selector 614 within the pixel predictor 604 determines that an intra-prediction, an inter-prediction, or interpolation operation is to be carried out. The predictor selector may furthermore output a predicted representation of an image block 616 to a first combiner 613. The predicted representation of the image block 616 is used in conjunction with the reconstructed prediction error signal 612 to generate a preliminary reconstructed image 618. The preliminary reconstructed image 618 may be used in the predictor 614 or may be passed to a filter 620. The filter 620 applies a filtering which outputs a final reconstructed signal 622. The final reconstructed signal 622 may be stored in a reference frame memory 624, the reference frame memory 624 further being connected to the predictor 614 for prediction operations.

The prediction error decoder 602 receives the output of the entropy decoder 600. A dequantizer 692 of the prediction error decoder 602 may dequantize the output of the entropy decoder 600 and the inverse transform block 693 may perform an inverse transform operation to the dequantized signal output by the dequantizer 692. The output of the entropy decoder 600 may also indicate that prediction error signal is not to be applied and in this case the prediction error decoder produces an all zero output signal.

The decoder selects the 16×16 pixel residual macroblock to reconstruct. The selection of the 16×16 pixel residual macroblock to be reconstructed is shown in step 700.

The decoder receives information on the encoding mode used when the current block has been encoded. The indication is decoded, when necessary, and provided to the reconstruction processor 691 of the prediction selector 614. The reconstruction processor 691 examines the indication (block 701 in FIG. 7) and selects one of the intra-prediction modes (block 703), if the indication indicates that the block has been encoded using intra-prediction, or an inter-prediction mode (blocks 704-711), if the indication indicates that the block has been encoded using inter-prediction.

If the current block has been encoded using inter-prediction, the pixel predictor 604 may operate as follows. The pixel predictor 604 receives motion vector information

16

(block 704). The pixel predictor 604 also receives (block 705) block type information and examines whether the block is a bi-predicted block or not (block 706). If the block type is a bi-predicted block, the pixel predictor 604 examines the motion vector information to determine which reference frames and reference block in the reference frames have been used in the construction of the motion vector information. The reconstruction processor 691 calculates the motion vectors (709) and uses the value of the (fraction of the) pixel of the reference blocks to which the motion vectors point to obtain a motion compensated prediction (710) and combines the prediction error with the value to obtain a reconstructed value of a pixel of the current block (block 711).

If the block type is a uni-predicted block, the pixel predictor 604 examines the motion vector information to determine which reference frame and reference block in the reference frame has been used in the construction of the motion vector information. The reconstruction processor 691 calculates the motion vector (707) and uses the value of the (fraction of the) pixel of the reference block to which the motion vector points to obtain a motion compensated prediction (708) and combines the prediction error with the value to obtain a reconstructed value of a pixel of the current block (block 711).

When the motion vector does not point to a full pixel sample in the reference block, the reconstruction processor 691 calculates using e.g. a one-directional interpolation or P-tap filtering (e.g. six-tap filtering) to obtain the values of the fractional pixels. Basically, the operations may be performed in the same way than in the encoder i.e. maintaining the higher accuracy values during the filtering until in the final rounding operation the accuracy may be decreased to the accuracy of the input pixels. Therefore, the effect of possible rounding errors may not be so large to the predicted values than in known methods.

The above described procedures may be repeated to each pixel of the current block to obtain all reconstructed pixel values for the current block.

In some embodiments the reconstruction processor 691 uses the interpolator 694 to perform the calculation of the fractional pixel values.

In some embodiments the reconstruction processor 691 provides the fractional pixel values to the predictor 695 which combines the fractional pixel values with prediction error to obtain the reconstructed values of the pixels of the current block.

In some embodiments the interpolation may also be performed by using full pixel values, half pixel values, and/or quarter pixel values which may have been stored into a reference frame memory. For example, the encoder or the decoder may comprise a reference frame memory in which the full pixel samples, half pixel values and quarter pixel values can be stored.

Furthermore, in some embodiments the type of the block may also be a multi-predicted block wherein the prediction of a block may be based on more than two reference blocks.

The embodiments of the invention described above describe the codec in terms of separate encoder and decoder apparatus in order to assist the understanding of the processes involved. However, it would be appreciated that the apparatus, structures and operations may be implemented as a single encoder-decoder apparatus/structure/operation. Furthermore in some embodiments of the invention the coder and decoder may share some or all common elements.

Although the above examples describe embodiments of the invention operating within a codec within an electronic device, it would be appreciated that the invention as

described below may be implemented as part of any video codec. Thus, for example, embodiments of the invention may be implemented in a video codec which may implement video coding over fixed or wired communication paths.

Thus, user equipment may comprise a video codec such as those described in embodiments of the invention above.

It shall be appreciated that the term user equipment is intended to cover any suitable type of wireless user equipment, such as mobile telephones, portable data processing devices or portable web browsers.

Furthermore elements of a public land mobile network (PLMN) may also comprise video codecs as described above.

In general, the various embodiments of the invention may be implemented in hardware or special purpose circuits, software, logic or any combination thereof. For example, some aspects may be implemented in hardware, while other aspects may be implemented in firmware or software which may be executed by a controller, microprocessor or other computing device, although the invention is not limited thereto. While various aspects of the invention may be illustrated and described as block diagrams, flow charts, or using some other pictorial representation, it is well understood that these blocks, apparatus, systems, techniques or methods described herein may be implemented in, as non-limiting examples, hardware, software, firmware, special purpose circuits or logic, general purpose hardware or controller or other computing devices, or some combination thereof.

The embodiments of this invention may be implemented by computer software executable by a data processor of the mobile device, such as in the processor entity, or by hardware, or by a combination of software and hardware. Further in this regard it should be noted that any blocks of the logic flow as in the Figures may represent program steps, or interconnected logic circuits, blocks and functions, or a combination of program steps and logic circuits, blocks and functions. The software may be stored on such physical media as memory chips, or memory blocks implemented within the processor, magnetic media such as hard disk or floppy disks, and optical media such as for example DVD and the data variants thereof, CD.

The memory may be of any type suitable to the local technical environment and may be implemented using any suitable data storage technology, such as semiconductor-based memory devices, magnetic memory devices and systems, optical memory devices and systems, fixed memory and removable memory. The data processors may be of any type suitable to the local technical environment, and may include one or more of general purpose computers, special purpose computers, microprocessors, digital signal processors (DSPs) and processors based on multi-core processor architecture, as non-limiting examples.

Embodiments of the inventions may be practiced in various components such as integrated circuit modules. The design of integrated circuits is by and large a highly automated process. Complex and powerful software tools are available for converting a logic level design into a semiconductor circuit design ready to be etched and formed on a semiconductor substrate.

Programs, such as those provided by Synopsys, Inc. of Mountain View, Calif. and Cadence Design, of San Jose, Calif. automatically route conductors and locate components on a semiconductor chip using well established rules of design as well as libraries of pre-stored design modules. Once the design for a semiconductor circuit has been completed, the resultant design, in a standardized electronic

format (e.g., Opus, GDSII, or the like) may be transmitted to a semiconductor fabrication facility or "fab" for fabrication.

The foregoing description has provided by way of exemplary and non-limiting examples a full and informative description of the exemplary embodiment of this invention. However, various modifications and adaptations may become apparent to those skilled in the relevant arts in view of the foregoing description, when read in conjunction with the accompanying drawings and the appended claims. However, all such and similar modifications of the teachings of this invention will still fall within the scope of this invention.

A method according to a first embodiment comprises:

determining a block of pixels of a video representation encoded in a bitstream, values of said pixels having a first precision;

determining a type of the block;

if the determining indicates that the block is a block predicted by using two or more reference blocks,

determining a first reference pixel location in a first reference block and a second reference pixel location in a second reference block;

using said first reference pixel location to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

using said second reference pixel location to obtain a second prediction, said second prediction having the second precision, which is higher than said first precision;

combining said first prediction and said second prediction to obtain a combined prediction; and

decreasing the precision of said combined prediction to said first precision.

In some methods according to the first embodiment a first rounding offset is inserted to said first prediction and said second prediction.

In some methods according to the first embodiment the precision of said first prediction and said second prediction is reduced to an intermediate prediction after adding said first rounding offset, said intermediate prediction being higher than said first precision.

In some methods according to the first embodiment a second rounding offset is inserted to the combined prediction before said decreasing.

In some methods according to the first embodiment said type of the block is a bi-directional block.

In some methods according to the first embodiment said type of the block is a multidirectional block.

In some methods according to the first embodiment the first rounding offset is 2^y , and said decreasing comprises right shifting the combined prediction $y+1$ bits.

In some methods according to the first embodiment the first precision is 8 bits.

In some methods according to the first embodiment the value of y is 5.

In some methods according to the first embodiment said first prediction and said second prediction are obtained by filtering pixel values of said reference blocks.

In some methods according to the first embodiment the filtering is performed by a P-tap filter.

An apparatus according to a second embodiment comprises:

a processor; and

a memory unit operatively connected to the processor and including:

computer code configured to determine a block of pixels of a video representation encoded in a bitstream, values of said pixels having a first precision;

computer code configured to determine a type of the block;

computer code configured to, if the determining indicates that the block is a block predicted by using two or more reference blocks,

determine a first reference pixel location in a first reference block and a second reference pixel location in a second reference block;

use said first reference pixel location to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

use said second reference pixel location to obtain a second prediction, said second prediction having the second precision, which is higher than said first precision;

combine said first prediction and said second prediction to obtain a combined prediction; and

decrease the precision of said combined prediction to said first precision.

In some apparatuses according to the second embodiment the computer code is further configured to insert a first rounding offset to said first prediction and said second prediction.

In some apparatuses according to the second embodiment the computer code is further configured to reduce the precision of said first prediction and said second prediction to an intermediate prediction after adding said first rounding offset, said intermediate prediction being higher than said first precision.

In some apparatuses according to the second embodiment the computer code is further configured to insert a second rounding offset to the combined prediction before said decreasing.

In some apparatuses according to the second embodiment said type of the block is a bi-directional block.

In some apparatuses according to the second embodiment said type of the block is a multidirectional block.

In some apparatuses according to the second embodiment the first rounding offset is 2^y , and said decreasing comprises right shifting the combined prediction $y+1$ bits.

In some apparatuses according to the second embodiment the first precision is 8 bits.

In some apparatuses according to the second embodiment the value of y is 5.

In some apparatuses according to the second embodiment the computer code is further configured to obtain said first prediction and said second prediction by filtering pixel values of said reference blocks.

In some apparatuses according to the second embodiment said filtering comprises a P-tap filter.

According to a third embodiment there is provided a computer readable storage medium stored with code thereon for use by an apparatus, which when executed by a processor, causes the apparatus to:

determine a block of pixels of a video representation encoded in a bitstream, values of said pixels having a first precision;

determine a type of the block;

if the determining indicates that the block is a block predicted by using two or more reference blocks, determine a first reference pixel location in a first reference block and a second reference pixel location in a second reference block;

use said first reference pixel location to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

use said second reference pixel location to obtain a second prediction, said second prediction having the second precision, which is higher than said first precision;

combine said first prediction and said second prediction to obtain a combined prediction; and

decrease the precision of said combined prediction to said first precision.

According to a fourth embodiment there is provided at least one processor and at least one memory, said at least one memory stored with code thereon, which when executed by said at least one processor, causes an apparatus to perform:

determine a block of pixels of a video representation encoded in a bitstream, values of said pixels having a first precision;

determine a type of the block;

if the determining indicates that the block is a block predicted by using two or more reference blocks,

determine a first reference pixel location in a first reference block and a second reference pixel location in a second reference block;

use said first reference pixel location to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

use said second reference pixel location to obtain a second prediction, said second prediction having the second precision, which is higher than said first precision;

combine said first prediction and said second prediction to obtain a combined prediction; and

decrease the precision of said combined prediction to said first precision.

According to some example embodiments the apparatus is an encoder.

According to some example embodiments the apparatus is a decoder.

An apparatus according to a fifth embodiment comprises: an input to determine a block of pixels of a video representation encoded in a bitstream, values of said pixels having a first precision;

a determinator to determine a type of the block; wherein if the determining indicates that the block is a block predicted by using two or more reference blocks, said determinator further to determine a first reference pixel location in a first reference block and a second reference pixel location in a second reference block;

a first predictor to use said first reference pixel location to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

a second predictor to use said second reference pixel location to obtain a second prediction, said second prediction having the second precision, which is higher than said first precision;

a combiner to combine said first prediction and said second prediction to obtain a combined prediction; and

a shifter to decrease the precision of said combined prediction to said first precision.

An apparatus according to a sixth embodiment comprises: means for determining a block of pixels of a video representation encoded in a bitstream, values of said pixels having a first precision;

means for determining a type of the block;

means for determining a first reference pixel location in a first reference block and a second reference pixel location in a second reference block, if the determining indicates that the block is a block predicted by using two or more reference blocks;

21

means for using said first reference pixel location to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

means for using said second reference pixel location to obtain a second prediction, said second prediction having the second precision, which is higher than said first precision;

means for combining said first prediction and said second prediction to obtain a combined prediction; and

means for decreasing the precision of said combined prediction to said first precision.

What is claimed is:

1. A method for encoding a block of pixels, the method comprising:

determining, for a current block, a first reference block based on a first motion vector and a second reference block based on a second motion vector, wherein the pixels of the current block, the first reference block, and the second reference block have values with a first precision;

using said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

using said second reference block to obtain a second prediction, said second prediction having the second precision;

obtaining a combined prediction based at least partly upon said first prediction and said second prediction;

decreasing a precision of said combined prediction by shifting bits of the combined prediction to the right; and encoding residual data in a bitstream, wherein the residual data is determined based upon a difference between the combined prediction and the block of pixels.

2. The method according to claim 1, wherein in an instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block.

3. The method according to claim 2, wherein said first prediction is obtained by interpolation using values of said first reference block by:

right shifting a sum of a P-tap filter using values of said first reference block.

4. The method according to claim 2, wherein in an instance in which said second motion vector points to an integer sample, said second prediction is obtained by shifting values of said second reference block to the left.

5. The method according to claim 1, wherein said decreasing said precision of said combined prediction by shifting bits of the combined prediction to the right, further comprises:

inserting a rounding offset to the combined prediction before said decreasing.

6. The method according to claim 1, wherein the first precision indicates a number of bits needed to represent the values of the pixels, and the second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction.

7. An apparatus for encoding a block of pixels, the apparatus comprising:

at least one processor and at least one memory including computer program code, the at least one memory and computer program code configured to, with the at least one processor, cause the apparatus to:

determine, for a current block, a first reference block based on a first motion vector and a second reference block based on a second motion vector, wherein the

22

pixels of the current block, the first reference block, and the second reference block have values with a first precision;

use said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

use said second reference block to obtain a second prediction, said second prediction having the second precision;

obtain a combined prediction based at least partly upon said first prediction and said second prediction;

decrease a precision of said combined prediction by shifting bits of the combined prediction to the right; and encode residual data in a bitstream, wherein the residual data is determined based upon a difference between the combined prediction and the block of pixels.

8. The apparatus according to claim 7, wherein in an instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block.

9. The apparatus according to claim 8, wherein said first prediction is obtained by interpolation using values of said first reference block by:

right shifting a sum of a P-tap filter using values of said first reference block.

10. The apparatus according to claim 8, wherein in an instance in which said second motion vector points to an integer sample, said second prediction is obtained by shifting values of said second reference block to the left.

11. The apparatus according to claim 7, wherein the at least one memory and computer code are configured to cause the apparatus to decrease said precision of said combined prediction by shifting bits of the combined prediction to the right, by:

inserting a rounding offset to the combined prediction before said decreasing.

12. The apparatus according to claim 7, wherein the first precision indicates a number of bits needed to represent the values of the pixels, and the second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction.

13. A computer program product for encoding a block of pixels, the computer program product comprising at least one non-transitory computer readable storage medium having computer executable program code portions stored therein, the computer executable program code portions comprising program code instructions configured to:

determine, for a current block, a first reference block based on a first motion vector and a second reference block based on a second motion vector, wherein the pixels of the current block, the first reference block, and the second reference block have values with a first precision;

use said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

use said second reference block to obtain a second prediction, said second prediction having the second precision;

obtain a combined prediction based at least partly upon said first prediction and said second prediction;

decrease a precision of said combined prediction by shifting bits of the combined prediction to the right; and encode residual data in a bitstream, wherein the residual data is determined based upon a difference between the combined prediction and the block of pixels.

23

14. The computer program product according to claim 13, wherein in an instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block.

15. The computer program product according to claim 14, wherein said first prediction is obtained by interpolation using values of said first reference block by:

right shifting a sum of a P-tap filter using values of said first reference block.

16. The computer program product according to claim 14, wherein in an instance in which said second motion vector points to an integer sample, said second prediction is obtained by shifting values of said second reference block to the left.

17. The computer program product according to claim 13, wherein the program code instructions configured to decrease said precision of said combined prediction by shifting bits of the combined prediction to the right, further comprise program code instructions configured to:

insert a rounding offset to the combined prediction before said decreasing.

18. The computer program product according to claim 13, wherein the first precision indicates a number of bits needed to represent the values of the pixels, and the second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction.

19. A method for decoding a block of pixels, the method comprising:

determining, for a current block, a first reference block based on a first motion vector and a second reference block based on a second motion vector, wherein the pixels of the current block, the first reference block, and the second reference block have values with a first precision;

using said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

using said second reference block to obtain a second prediction, said second prediction having the second precision;

obtaining a combined prediction based at least partly upon said first prediction and said second prediction;

decreasing a precision of said combined prediction by shifting bits of the combined prediction to the right; and reconstructing the block of pixels based on the combined prediction.

20. The method according to claim 19, wherein in an instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block.

21. The method according to claim 20, wherein said first prediction is obtained by interpolation using values of said first reference block by:

right shifting a sum of a P-tap filter using values of said first reference block.

22. The method according to claim 20, wherein in an instance in which said second motion vector points to an integer sample, said second prediction is obtained by shifting values of said second reference block to the left.

23. The method according to claim 19, wherein said decreasing said precision of said combined prediction by shifting bits of the combined prediction to the right, further comprises:

inserting a rounding offset to the combined prediction before said decreasing.

24. The method according to claim 19, wherein the first precision indicates a number of bits needed to represent the

24

values of the pixels, and the second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction.

25. An apparatus for decoding a block of pixels, the apparatus comprising:

at least one processor and at least one memory including computer program code, the at least one memory and computer program code configured to, with the at least one processor, cause the apparatus to:

determine, for a current block, a first reference block based on a first motion vector and a second reference block based on a second motion vector, wherein the pixels of the current block, the first reference block, and the second reference block have values with a first precision;

use said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

use said second reference block to obtain a second prediction, said second prediction having the second precision;

obtain a combined prediction based at least partly upon said first prediction and said second prediction;

decrease a precision of said combined prediction by shifting bits of the combined prediction to the right; and reconstruct the block of pixels based on the combined prediction.

26. The apparatus according to claim 25, wherein in an instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block.

27. The apparatus according to claim 26, wherein said first prediction is obtained by interpolation using values of said first reference block by:

right shifting a sum of a P-tap filter using values of said first reference block.

28. The apparatus according to claim 26, wherein in an instance in which said second motion vector points to an integer sample, said second prediction is obtained by shifting values of said second reference block to the left.

29. The apparatus according to claim 25, wherein the at least one memory and computer code are configured to cause the apparatus to decrease said precision of said combined prediction by shifting bits of the combined prediction to the right, by:

inserting a rounding offset to the combined prediction before said decreasing.

30. The apparatus according to claim 25, wherein the first precision indicates a number of bits needed to represent the values of the pixels, and the second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction.

31. A computer program product for decoding a block of pixels, the computer program product comprising at least one non-transitory computer readable storage medium having computer executable program code portions stored therein, the computer executable program code portions comprising program code instructions configured to:

determine, for a current block, a first reference block based on a first motion vector and a second reference block based on a second motion vector, wherein the pixels of the current block, the first reference block, and the second reference block have values with a first precision;

use said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

25

use said second reference block to obtain a second prediction, said second prediction having the second precision;

obtain a combined prediction based at least partly upon said first prediction and said second prediction;

decrease a precision of said combined prediction by shifting bits of the combined prediction to the right; and reconstruct the block of pixels based on the combined prediction.

32. The computer program product according to claim **31**, wherein in an instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block.

33. The computer program product according to claim **32**, wherein said first prediction is obtained by interpolation using values of said first reference block by:

right shifting a sum of a P-tap filter using values of said first reference block.

26

34. The computer program product according to claim **32**, wherein in an instance in which said second motion vector points to an integer sample, said second prediction is obtained by shifting values of said second reference block to the left.

35. The computer program product according to claim **31**, wherein the program code instructions configured to decrease said precision of said combined prediction by shifting bits of the combined prediction to the right, further comprise program code instructions configured to:

insert a rounding offset to the combined prediction before said decreasing.

36. The computer program product according to claim **31**, wherein the first precision indicates a number of bits needed to represent the values of the pixels, and the second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction.

* * * * *

EXHIBIT 18



US009390137B2

(12) **United States Patent**
Setlur et al.

(10) **Patent No.:** **US 9,390,137 B2**
(45) **Date of Patent:** **Jul. 12, 2016**

(54) **METHOD AND APPARATUS FOR PROVIDING AN ORDERING METRIC FOR A MULTI-DIMENSIONAL CONTEXTUAL QUERY**

(75) Inventors: **Vidya Setlur**, Portola Valley, CA (US);
Agathe Battestini, San Francisco, CA (US)

(73) Assignee: **NOKIA TECHNOLOGIES OY**, Espoo (FI)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 764 days.

(21) Appl. No.: **13/172,425**

(22) Filed: **Jun. 29, 2011**

(65) **Prior Publication Data**

US 2013/0007011 A1 Jan. 3, 2013

(51) **Int. Cl.**

G06F 7/00 (2006.01)

G06F 17/00 (2006.01)

G06F 17/30 (2006.01)

(52) **U.S. Cl.**

CPC **G06F 17/30528** (2013.01)

(58) **Field of Classification Search**

USPC 707/732, 726, 791; 706/45, 62; 715/811
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,693,651 B2 2/2004 Biebesheimer et al.
7,330,849 B2 * 2/2008 Gerasoulis et al.
7,747,618 B2 * 6/2010 Zeng et al. 707/732
7,853,574 B2 12/2010 Kraenzel et al.
2002/0055924 A1 5/2002 Liming
2002/0188589 A1 12/2002 Salmenkaita et al.
2003/0036848 A1 2/2003 Sheha et al.

2004/0110515 A1 6/2004 Blumberg et al.
2004/0162830 A1 8/2004 Shirwadkar et al.
2008/0301092 A1 12/2008 Jayanti et al.
2009/0177386 A1 7/2009 Haase
2009/0228196 A1 9/2009 Raab
2010/0094904 A1 * 4/2010 Jandhyala 707/791
2010/0145976 A1 6/2010 Higgins et al.
2010/0223261 A1 * 9/2010 Sarkar 707/726
2010/0241663 A1 9/2010 Huang et al.

(Continued)

FOREIGN PATENT DOCUMENTS

KR 2007-0061024 A 6/2007
WO WO 01/82556 A2 11/2001
WO WO 01/82562 A2 11/2001
WO WO 2010/016989 A2 2/2010
WO WO 2010/039459 A2 4/2010

OTHER PUBLICATIONS

International Search Report for PCT/FI2012/050468 dated Sep. 6, 2012, pp. 1-6.

(Continued)

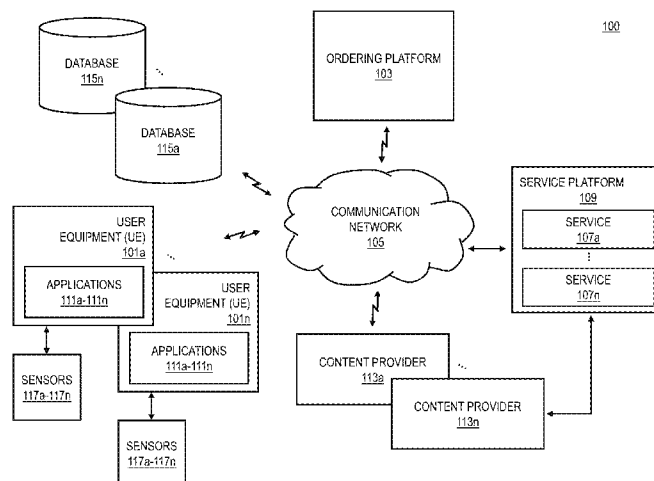
Primary Examiner — Susan Chen

(74) *Attorney, Agent, or Firm* — Dithavong & Steiner, P.C.

(57) **ABSTRACT**

An approach is provided for providing an ordering metric for a multi-dimensional contextual query. An ordering platform determines a multi-dimensional query associated with at least one user device, wherein the multi-dimensional query specifies, at least in part, one or more personas, one or more contexts, or a combination thereof associated with the at least one user device. The ordering platform further causes, at least in part, an execution of the multi-dimensional query on at least one context-sensitive database to generate one or more results. The ordering platform further determines at least one ordering metric for the one or more results based, at least in part, on one or more user contextual attributes of the at least one user device.

20 Claims, 11 Drawing Sheets



(56)

References Cited

OTHER PUBLICATIONS

U.S. PATENT DOCUMENTS

2011/0055203 A1 3/2011 Gutt et al.

2011/0078159 A1 * 3/2011 Li et al. 707/749

Written Opinion for PCT/FI2012/050468 dated Sep. 6, 2012, pp. 1-9.

* cited by examiner

FIG. 1

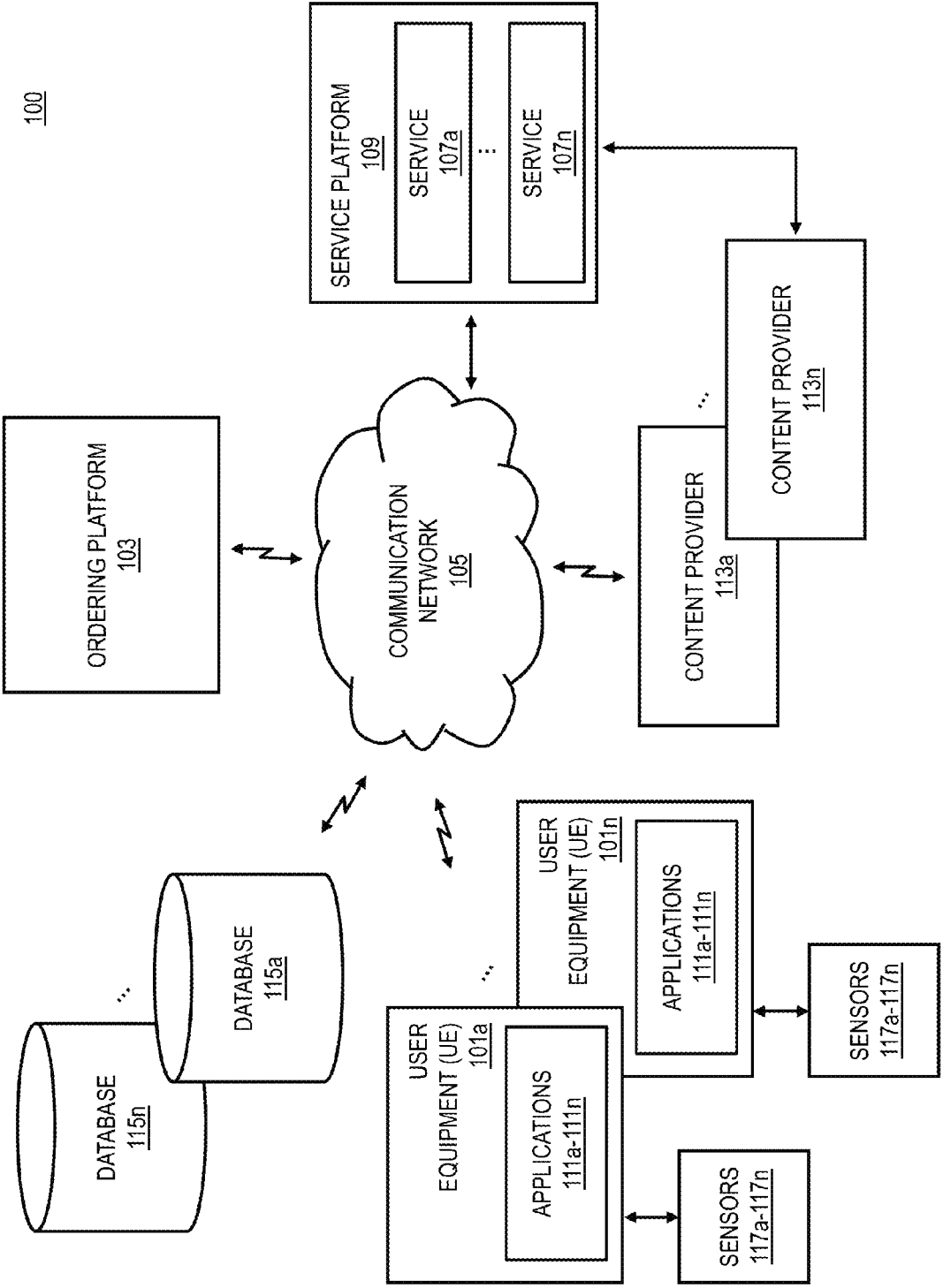


FIG. 2

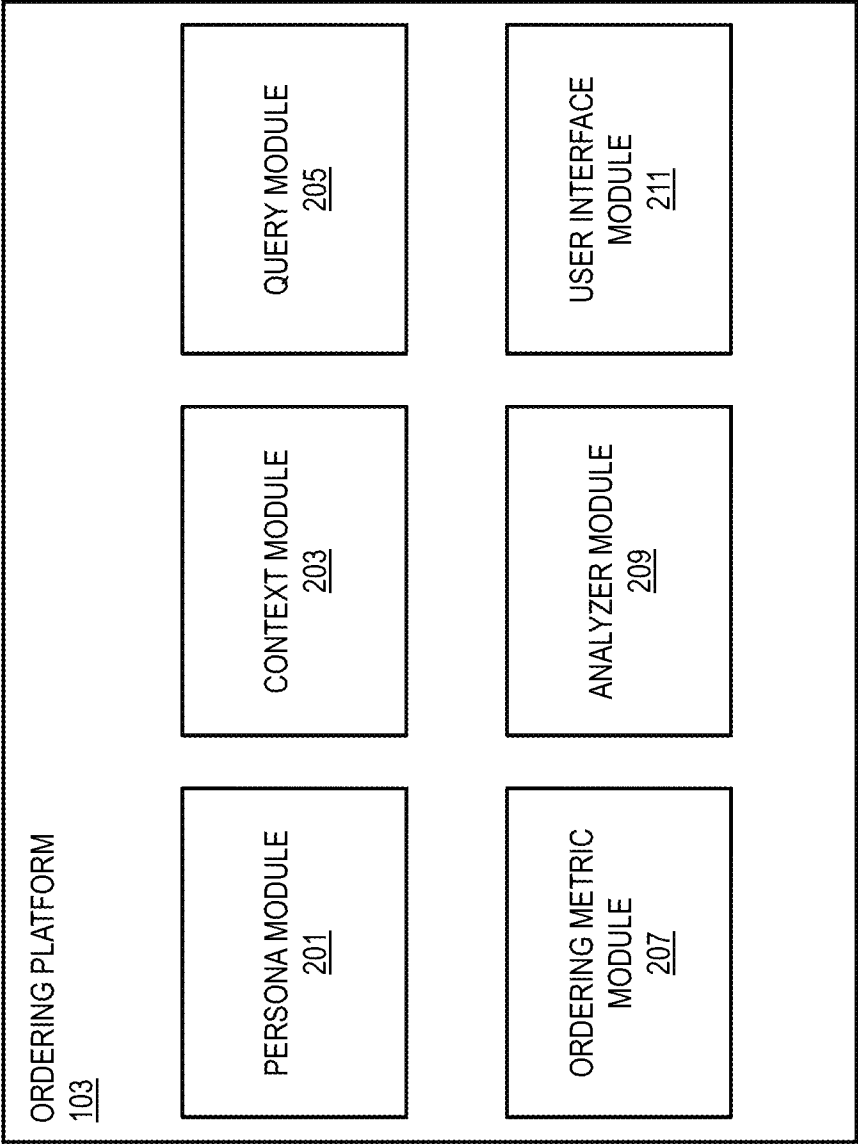


FIG. 3

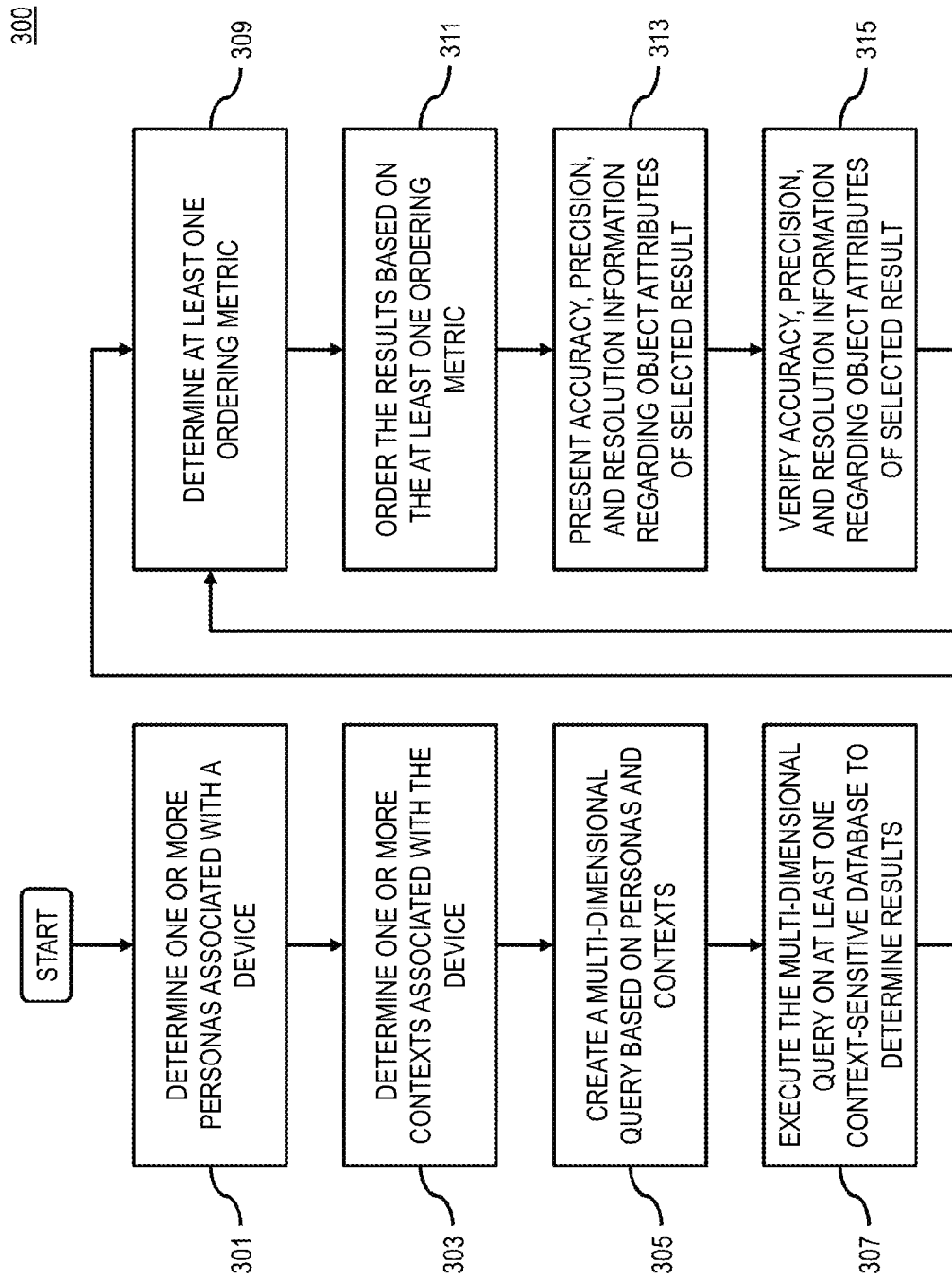


FIG. 4A

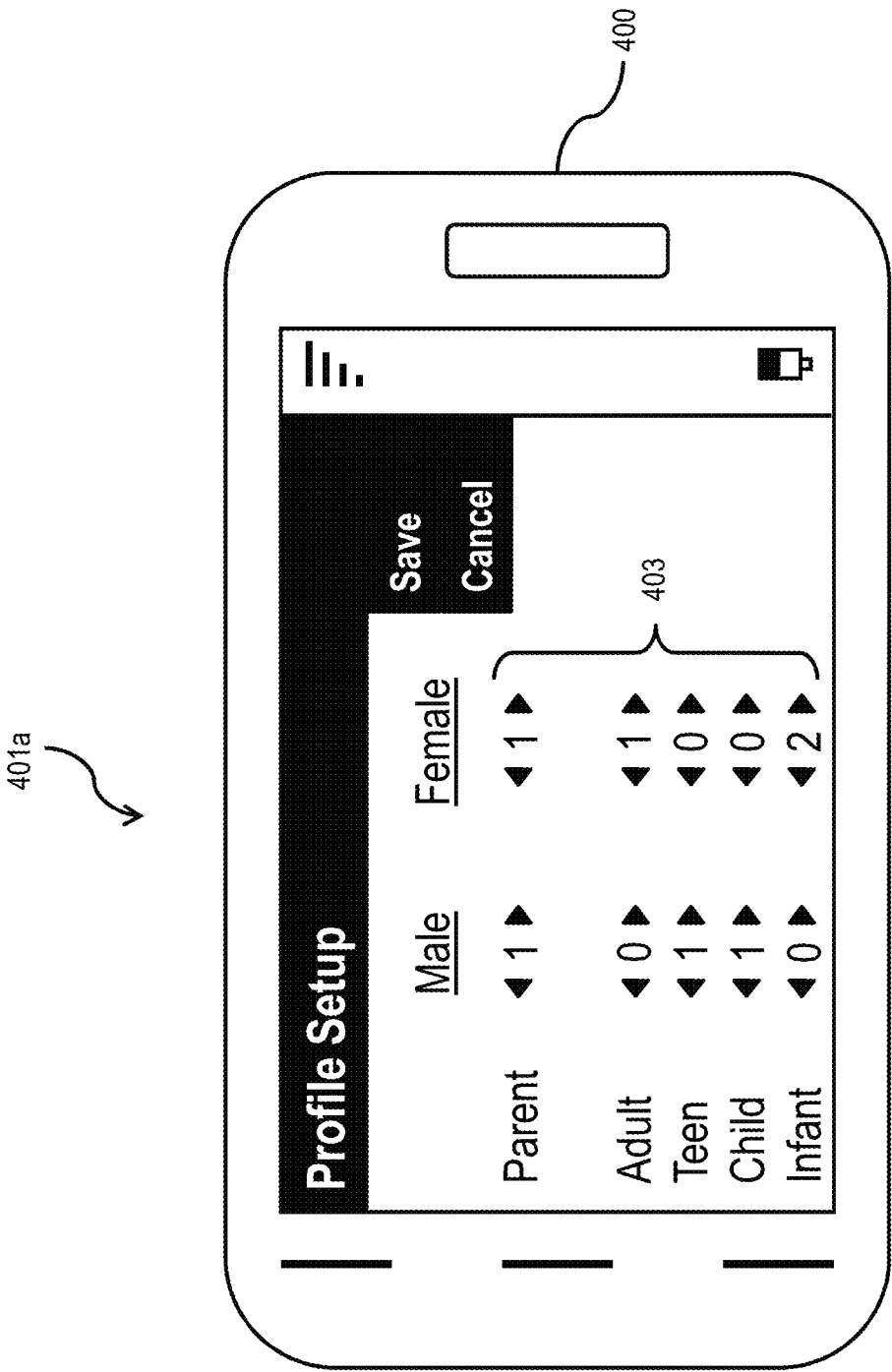


FIG. 4B

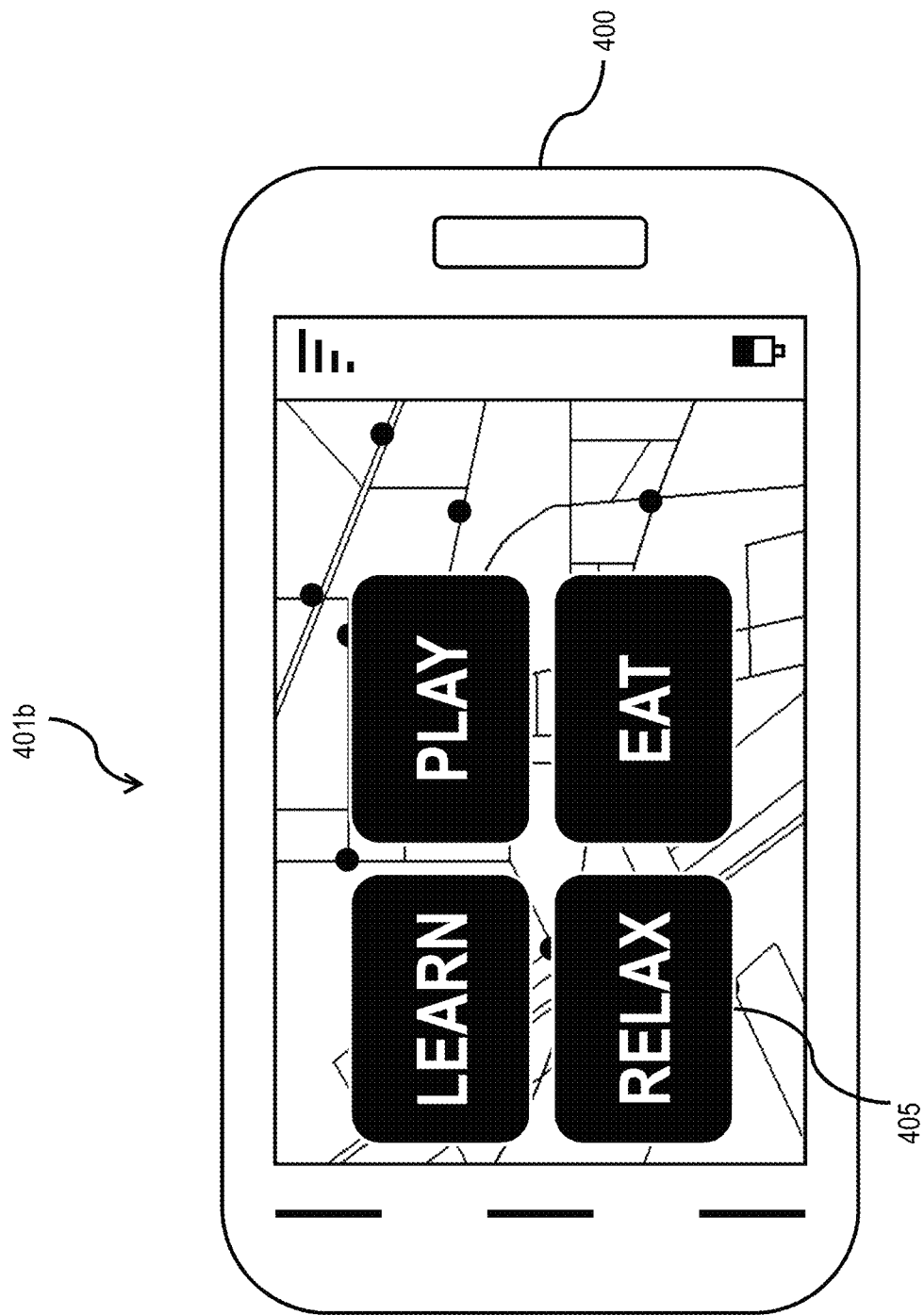


FIG. 4C

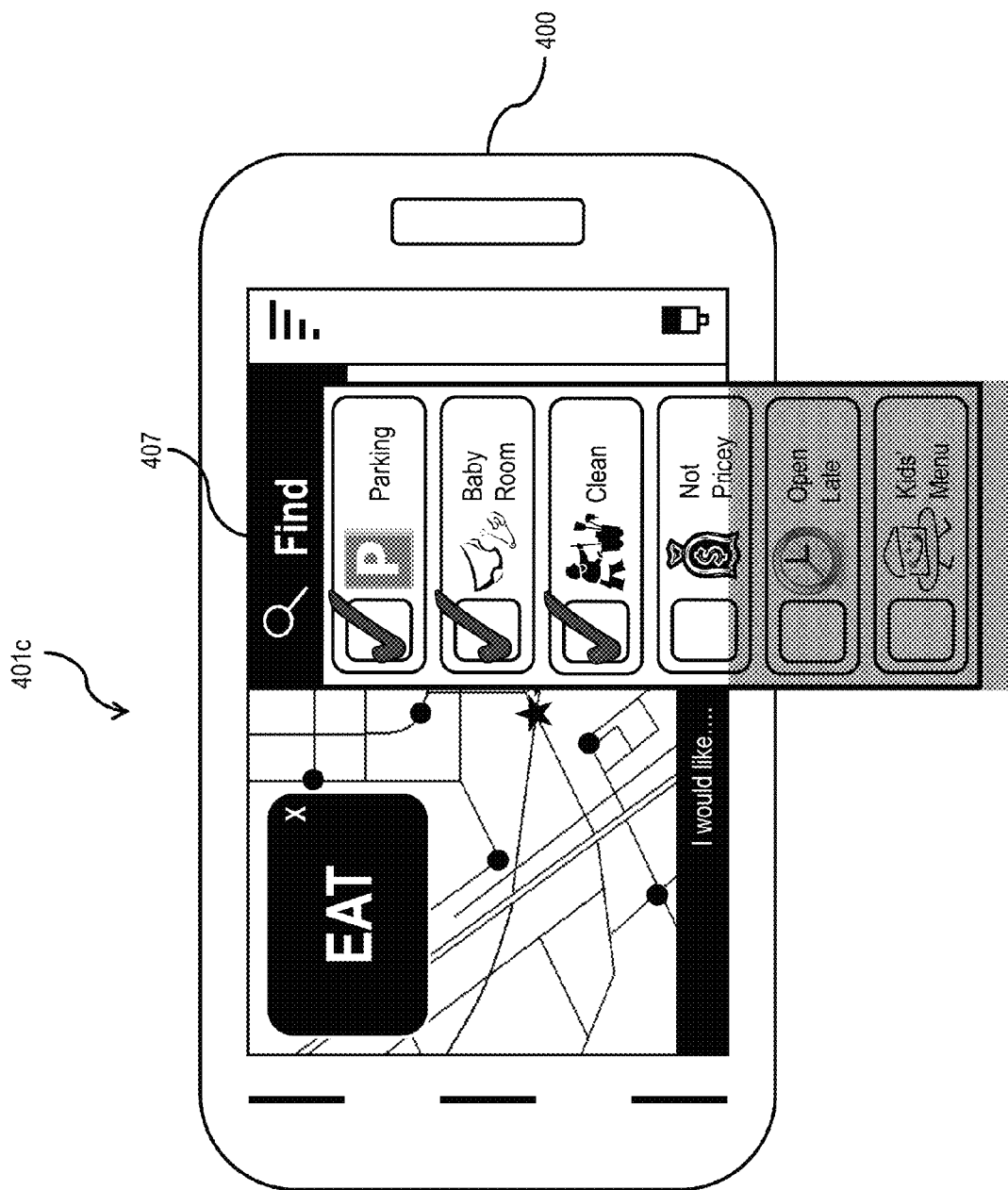


FIG. 4D

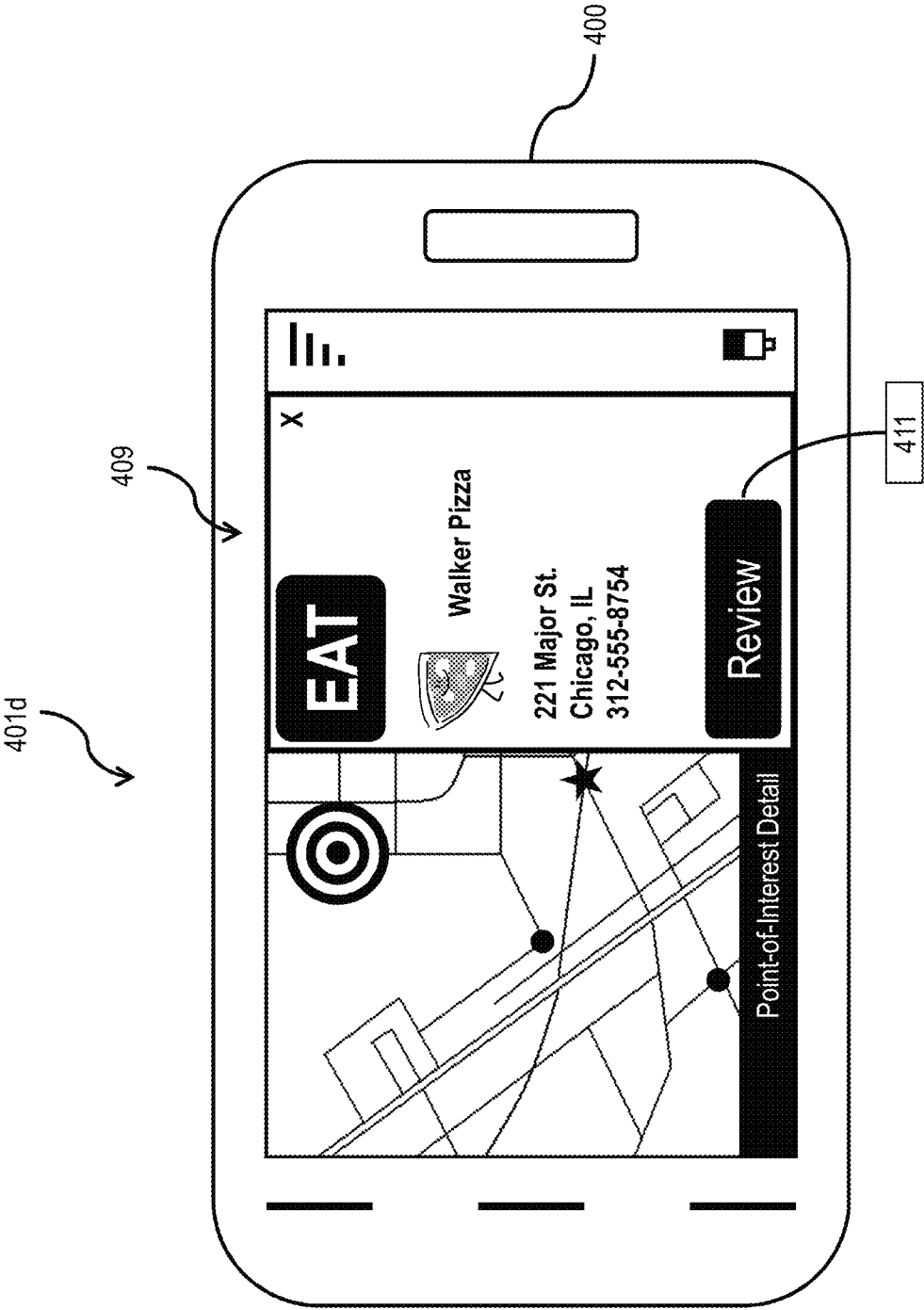


FIG. 4E

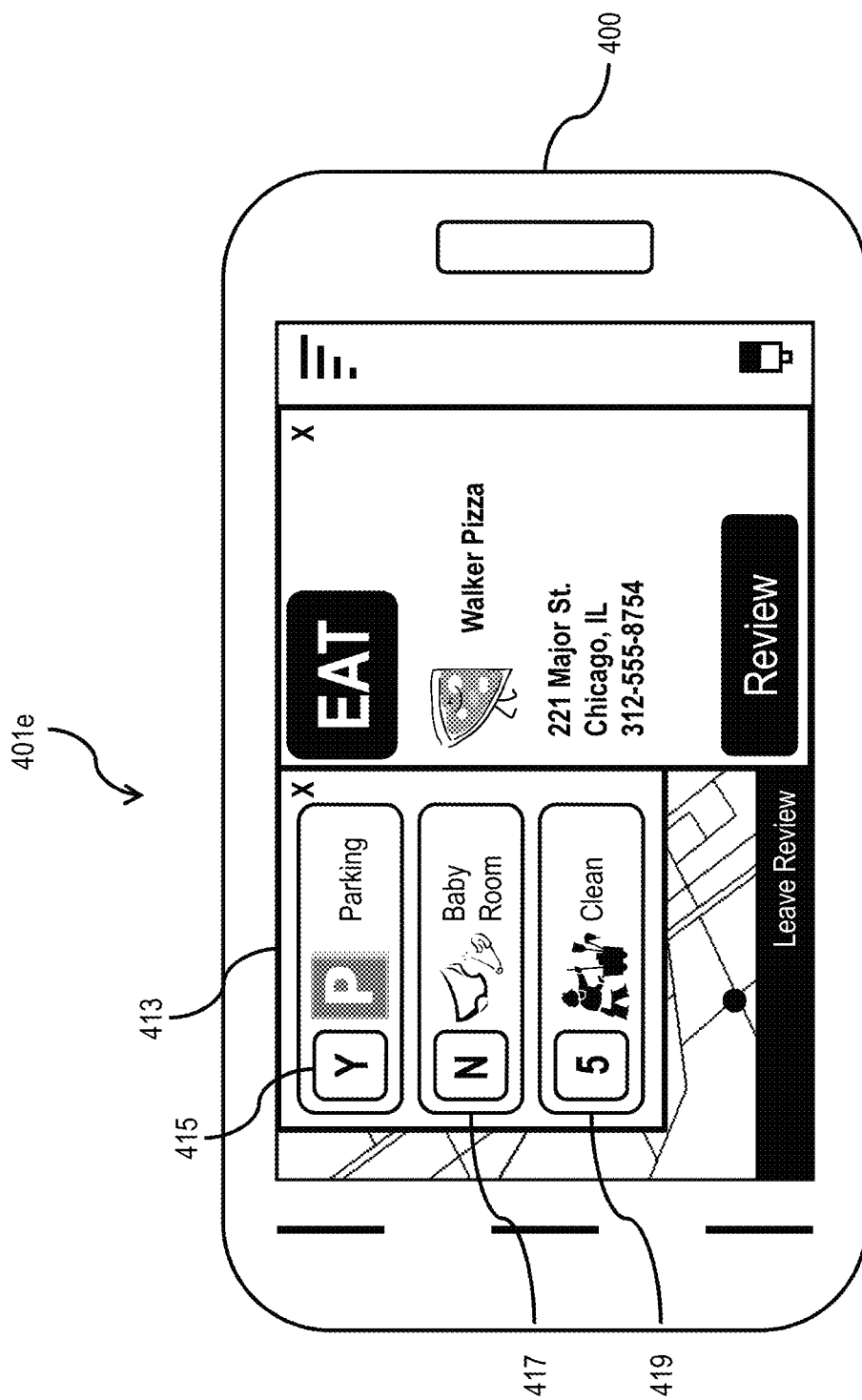


FIG. 5

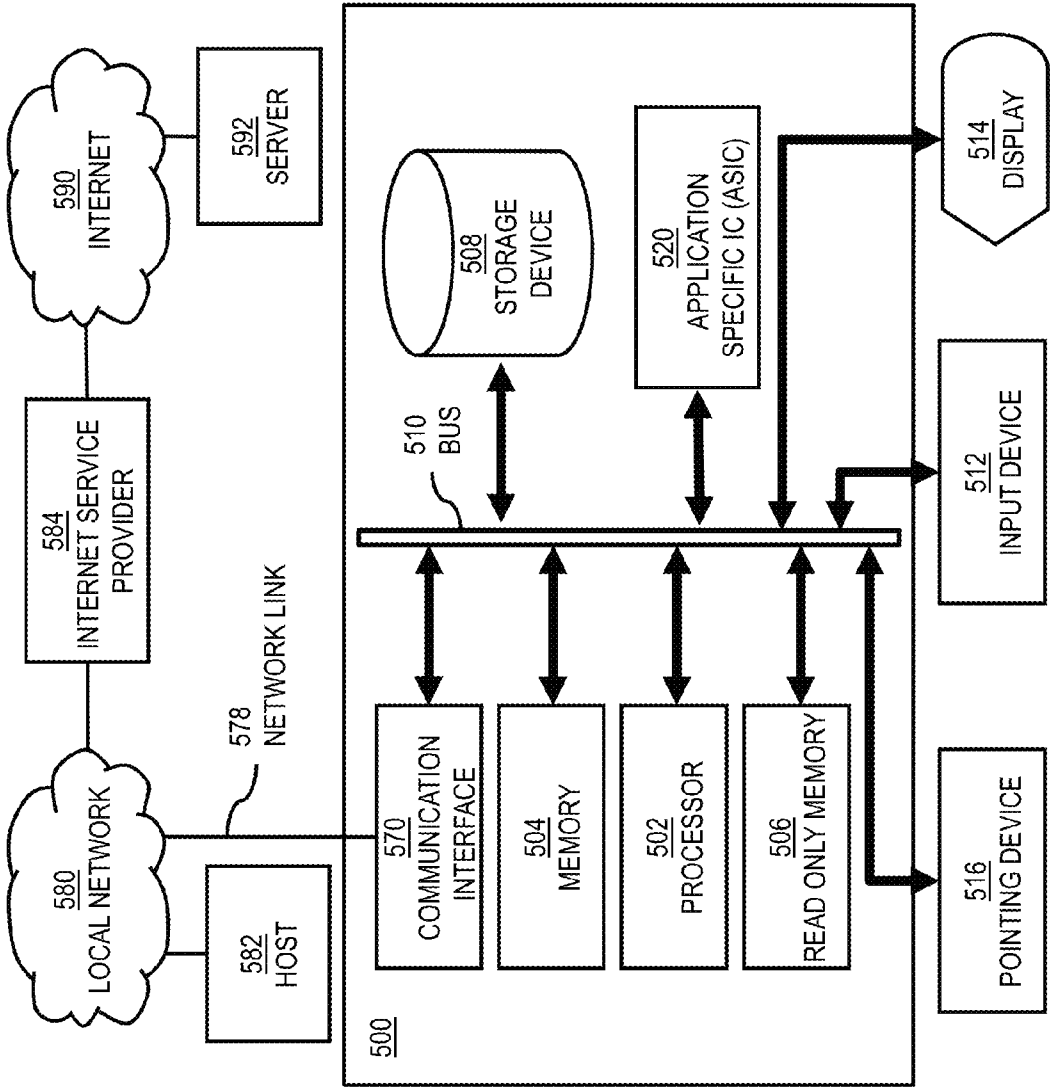


FIG. 6

600

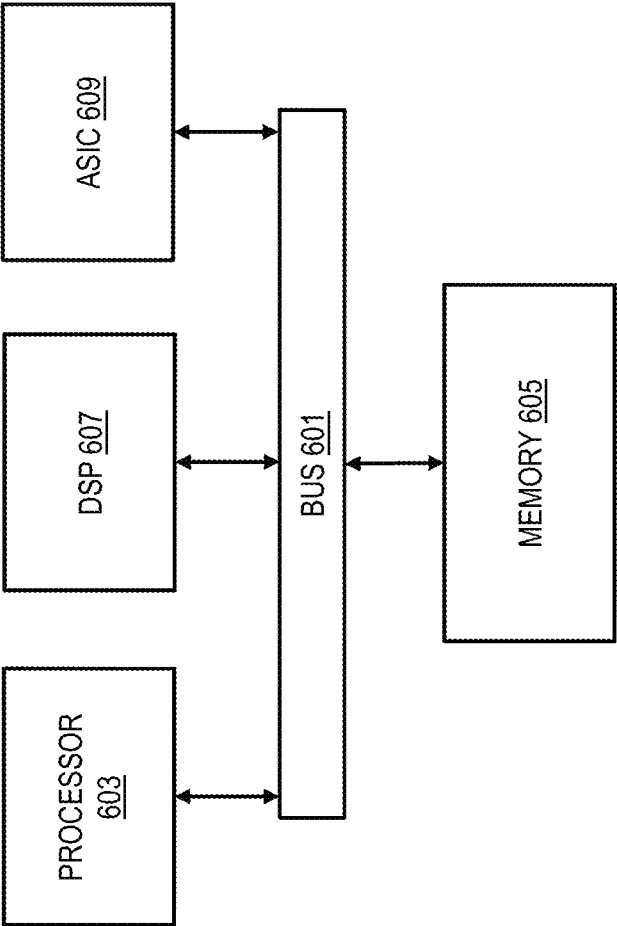
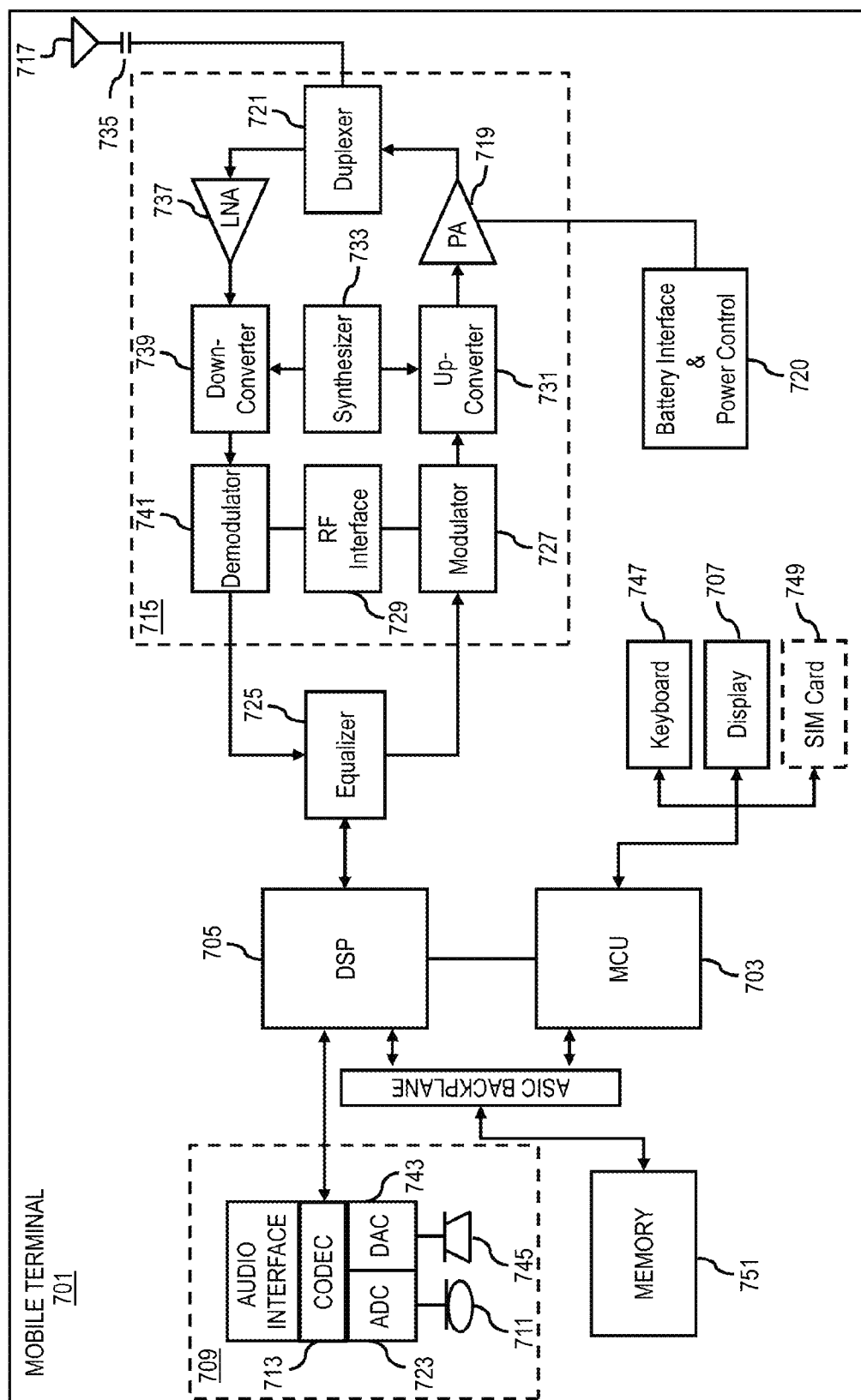


FIG. 7



1

METHOD AND APPARATUS FOR PROVIDING AN ORDERING METRIC FOR A MULTI-DIMENSIONAL CONTEXTUAL QUERY

BACKGROUND

Service providers and device manufacturers (e.g., wireless, cellular, etc.) are continually challenged to deliver value and convenience to consumers by, for example, providing compelling network services and relevant content. Often, such network and content services rely on the context of the users accessing the network services or the devices by which the users access the network services. Determining the context becomes inadequate when multiple contexts and/or multiple users' actions need to be analyzed over a period of time—particularly when the context is determined in conjunction with a mobile device that has limited processing power and/or bandwidth. However, there still exists a need to have rich, accurate datasets that span different context-aware situations to allow applications that utilize the datasets to be useful for multiple contexts and/or multiple users and provide access to such datasets to mobile devices in a practical manner. As such, device manufacturers and service providers face significant technical challenges to providing rich, situational-aware, context-sensitive datasets in a mobile environment.

SOME EXAMPLE EMBODIMENTS

Therefore, there is a need for an approach for providing an ordering metric for a multi-dimensional contextual query to provide context-aware querying, retrieval and presentation of results on, for example, a mobile interface.

According to one embodiment, a method comprises determining a multi-dimensional query associated with at least one user device, wherein the multi-dimensional query specifies, at least in part, one or more personas, one or more contexts, or a combination thereof associated with the at least one user device. The method also comprises causing, at least in part, an execution of the multi-dimensional query on at least one context-sensitive database to generate one or more results. The method further comprises determining at least one ordering metric for the one or more results based, at least in part, on one or more user contextual attributes of the at least one user device.

According to another embodiment, an apparatus comprises at least one processor, and at least one memory including computer program code for one or more computer programs, the at least one memory and the computer program code configured to, with the at least one processor, cause, at least in part, the apparatus to determine a multi-dimensional query associated with at least one user device, wherein the multi-dimensional query specifies, at least in part, one or more personas, one or more contexts, or a combination thereof associated with the at least one user device. The apparatus is also caused, at least in part, to execute the multi-dimensional query on at least one context-sensitive database to generate one or more results. The apparatus is further caused to determine at least one ordering metric for the one or more results based, at least in part, on one or more user contextual attributes of the at least one user device.

According to another embodiment, a computer-readable storage medium carries one or more sequences of one or more instructions which, when executed by one or more processors, cause, at least in part, an apparatus to determine a multi-dimensional query associated with at least one user device, wherein the multi-dimensional query specifies, at least in

2

part, one or more personas, one or more contexts, or a combination thereof associated with the at least one user device. The apparatus is also caused, at least in part, to execute the multi-dimensional query on at least one context-sensitive database to generate one or more results. The apparatus is further caused to determine at least one ordering metric for the one or more results based, at least in part, on one or more user contextual attributes of the at least one user device.

According to another embodiment, an apparatus comprises means for determining a multi-dimensional query associated with at least one user device, wherein the multi-dimensional query specifies, at least in part, one or more personas, one or more contexts, or a combination thereof associated with the at least one user device. The apparatus also comprises means for causing, at least in part, an execution of the multi-dimensional query on at least one context-sensitive database to generate one or more results. The apparatus further comprises means for determining at least one ordering metric for the one or more results based, at least in part, on one or more user contextual attributes of the at least one user device.

In addition, for various example embodiments of the invention, the following is applicable: a method comprising facilitating a processing of and/or processing (1) data and/or (2) information and/or (3) at least one signal, the (1) data and/or (2) information and/or (3) at least one signal based, at least in part, on (or derived at least in part from) any one or any combination of methods (or processes) disclosed in this application as relevant to any embodiment of the invention.

For various example embodiments of the invention, the following is also applicable: a method comprising facilitating access to at least one interface configured to allow access to at least one service, the at least one service configured to perform any one or any combination of network or service provider methods (or processes) disclosed in this application.

For various example embodiments of the invention, the following is also applicable: a method comprising facilitating creating and/or facilitating modifying (1) at least one device user interface element and/or (2) at least one device user interface functionality, the (1) at least one device user interface element and/or (2) at least one device user interface functionality based, at least in part, on data and/or information resulting from one or any combination of methods or processes disclosed in this application as relevant to any embodiment of the invention, and/or at least one signal resulting from one or any combination of methods (or processes) disclosed in this application as relevant to any embodiment of the invention.

For various example embodiments of the invention, the following is also applicable: a method comprising creating and/or modifying (1) at least one device user interface element and/or (2) at least one device user interface functionality, the (1) at least one device user interface element and/or (2) at least one device user interface functionality based at least in part on data and/or information resulting from one or any combination of methods (or processes) disclosed in this application as relevant to any embodiment of the invention, and/or at least one signal resulting from one or any combination of methods (or processes) disclosed in this application as relevant to any embodiment of the invention.

In various example embodiments, the methods (or processes) can be accomplished on the service provider side or on the mobile device side or in any shared way between service provider and mobile device with actions being performed on both sides.

For various example embodiments, the following is applicable: An apparatus comprising means for performing the method of any of the claims.

3

Still other aspects, features, and advantages of the invention are readily apparent from the following detailed description, simply by illustrating a number of particular embodiments and implementations, including the best mode contemplated for carrying out the invention. The invention is also capable of other and different embodiments, and its several details can be modified in various obvious respects, all without departing from the spirit and scope of the invention. Accordingly, the drawings and description are to be regarded as illustrative in nature, and not as restrictive.

BRIEF DESCRIPTION OF THE DRAWINGS

The embodiments of the invention are illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings:

FIG. 1 is a diagram of a system capable of providing an ordering metric for a multi-dimensional contextual query, according to one embodiment;

FIG. 2 is a diagram of the components of an ordering platform, according to one embodiment;

FIG. 3 is a flowchart of a process for providing an ordering metric for a multi-dimensional contextual query, according to one embodiment;

FIGS. 4A-4E are diagrams of user interfaces utilized in the process of FIG. 3, according to various embodiments;

FIG. 5 is a diagram of hardware that can be used to implement an embodiment of the invention;

FIG. 6 is a diagram of a chip set that can be used to implement an embodiment of the invention; and

FIG. 7 is a diagram of a mobile terminal (e.g., handset) that can be used to implement an embodiment of the invention.

DESCRIPTION OF SOME EMBODIMENTS

Examples of a method, apparatus, and computer program for providing an ordering metric for a multi-dimensional contextual query are disclosed. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the embodiments of the invention. It is apparent, however, to one skilled in the art that the embodiments of the invention may be practiced without these specific details or with an equivalent arrangement. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the embodiments of the invention.

Although various embodiments are described with respect to context-aware databases storing points of interest, it is contemplated that the approach described herein may be used with other context-aware databases storing any type of contextually based information.

FIG. 1 is a diagram of a system capable of providing an ordering metric for a multi-dimensional contextual query, according to one embodiment. As discussed above, services and/or content provided over a network to users' devices often rely on the current context of the users and/or the devices. When multiple contexts and/or multiple users' actions need to be analyzed over a period of time, adapting the applications to use the contexts and/or actions becomes inadequate. Although situation awareness can correlate specific contexts and user actions to specific contextual situations, using such situation awareness has not been incorporated into context-aware databases. Thus, there still exists a need to have rich, accurate datasets that span different context-aware situations to allow applications that utilize the datasets to be useful for multiple contexts and/or multiple users.

4

To address this problem, a system 100 of FIG. 1 introduces the capability to enrich and enhance situational-aware, context-sensitive databases using mechanisms on, for example, a mobile device. The system 100 also has the framework for allowing the mobile user to provide additional information concerning relevant attributes for a data entity in a context-aware database. The system 100 also introduces context-based attribute ordering in the interface of, for example, a mobile device to provide more fluid and less intensive context-aware querying, retrieval and presentation of contextual attributes of objects stored in databases.

As shown in FIG. 1, the system 100 comprises user equipment (UE) 101a-101n (collectively referred to as UE 101) having connectivity to an ordering platform 103 (discussed in detail below) via a communication network 105. The UE 101 include one or more executable applications 111a-111n (collectively referred to as applications 111) that include, for example, one or more mapping applications, messaging applications, calendar applications, context applications, sensor applications, etc. In communication with the UE 101 (e.g., within the UE 101, outside the UE 101) are one or more sensors 117a-117n (collectively referred to as sensors 117) that acquire information regarding the context of the user and/or the UE 101. The sensors 117 can collect any type of information depending on the type of sensor. For example, the sensors 117 can constitute accelerometers, gyroscopes, brightness sensors, moisture sensors, etc. that collect information regarding acceleration, light and water vapor of the UE 101 and the surrounding environment.

In communication with the ordering platform 103 and the UE 101 via the communication network 105 are databases 115a-115n (collectively referred to as databases 115). The databases 115 can be 2-dimensional, relational databases or multi-dimensional (e.g., 3-dimensional, N-dimensional, etc.) databases that are organized into hierarchies that specify an aggregation level and granularity of viewing data.

The databases store objects and information concerning the objects, such as object contextual attributes. The object contextual attributes correspond to the context of the objects, such as, for example, the geographic location of the objects (e.g., address, city, country, etc.), the type of object (e.g., restaurants, schools, parks, and playgrounds), and all other possible types of contextual information related to the objects. By way of example, for an object such as a park, the additional contextual attributes can include whether the park has a playground, whether the park has restrooms, whether the restrooms are clean, whether the park has food concessions, whether the food concessions have a wide range of choices, whether the park has a parking lot or street parking, whether access to the park is free, etc. Each one of these contextual attributes can be further defined by the accuracy, the precision and the resolution of the contextual attributes. For example, the contextual attributes regarding whether the restrooms are clean and/or whether the food concessions have a wide range of choices can have additional attributes indicating the accuracy, the precision and the resolution of the cleanliness of the concessions or the range of choices.

Attributes representing the accuracy, the precision and the resolution can be provided by the provider of the system 100, by the users of the system 100, or a combination thereof. For example, initially the values can be provided by the provider of the system 100 and can be modified, updated, or removed by the users of the system 100. The system 100 can also combine attributes of any one or more of the accuracy, the precision and the resolution into one, singular attribute. For instance, if the information about the cleanliness of the restrooms is both accurate and precise, a single attribute may

5

correspond to “very clean.” If the information about the cleanliness is accurate but not precise, the attribute may correspond to a range, such as “not clean-clean.” In other instances, if the information regarding the cleanliness of the restroom is based on parts of the restroom, the system **100** may indicate the resolution of the information by providing an overall attribute of the cleanliness of the restroom, and also provide discrete attributes for the cleanliness of, for example, the sink, the toilet, the towels etc. within the restroom.

For multi-dimensional databases, each dimension is defined over a dimensional schema, which is a set of dimensional attributes. By way of example, one multidimensional schema can be user preferences attributes. The dimensional attributes of user preferences are, for example, users, location based preferences, context based preferences, behavior based preferences, etc. In this example, the user dimensional attribute is terminal and belongs to the first category level, the location based preferences belongs to the second category level, the context based preferences belongs to the third category level, the behavior based preferences belongs to the fourth category level, etc. By way of further example, each of these dimensional attributes can have further nested multi-dimensional attributes. For example, the location based preferences can include current location based preferences, nearby location based preferences, far location based preferences, etc. where each of these specific location based preferences are representative of a smaller granularity of the location.

The system **100** also includes a service platform **109** that provides one or more services **107a-107n** (collectively known as services **107**) to one or more users and/or user devices. The services **107** can include, for example, location based services, mapping information services, social networking services, etc. In one embodiment, the service platform **109** includes a service **107b** that can convert a 2-dimensional, relational database into a multi-dimensional database for use with the multi-dimensional query of the system **100** according to well-known conversion techniques.

Also included within the system **100** is one or more content providers **113a-113n** (collectively referred to as content providers **113**) that can provide content to one or more services **107** of the service platform **109**, one or more UE **101**, one or more databases **115** and/or the ordering platform **103**. By way of example, as the information regarding the objects stored in the databases **115** changes, the content providers **113** can update the information within the databases. When new objects are created or removed, such as when a new point of interest is created or a new point of interest is closed, the content providers **113** can update the new information within the databases **115**.

By way of example, the communication network **105** of the system **100** includes one or more networks such as a data network, a wireless network, a telephony network, or any combination thereof. It is contemplated that the data network may be any local area network (LAN), metropolitan area network (MAN), wide area network (WAN), a public data network (e.g., the Internet), short range wireless network, or any other suitable packet-switched network, such as a commercially owned, proprietary packet-switched network, e.g., a proprietary cable or fiber-optic network, and the like, or any combination thereof. In addition, the wireless network may be, for example, a cellular network and may employ various technologies including enhanced data rates for global evolution (EDGE), general packet radio service (GPRS), global system for mobile communications (GSM), Internet protocol multimedia subsystem (IMS), universal mobile telecommunications system (UMTS), etc., as well as any other suitable

6

wireless medium, e.g., worldwide interoperability for microwave access (WiMAX), Long Term Evolution (LTE) networks, code division multiple access (CDMA), wideband code division multiple access (WCDMA), wireless fidelity (WiFi), wireless LAN (WLAN), Bluetooth®, Internet Protocol (IP) data casting, satellite, mobile ad-hoc network (MANET), and the like, or any combination thereof.

The UE **101** is any type of mobile terminal, fixed terminal, or portable terminal including a mobile handset, station, unit, device, multimedia computer, multimedia tablet, Internet node, communicator, desktop computer, laptop computer, notebook computer, netbook computer, tablet computer, personal communication system (PCS) device, personal navigation device, personal digital assistants (PDAs), audio/video player, digital camera/camcorder, positioning device, television receiver, radio broadcast receiver, electronic book device, game device, or any combination thereof, including the accessories and peripherals of these devices, or any combination thereof. It is also contemplated that the UE **101** can support any type of interface to the user (such as “wearable” circuitry, etc.).

By way of example, the UE **101**, the ordering platform **103**, the service platform **109**, the content providers **113** and the databases **115** communicate with each other and other components of the communication network **105** using well known, new and/or still developing protocols. In this context, a protocol includes a set of rules defining how the network nodes within the communication network **105** interact with each other based on information sent over the communication links. The protocols are effective at different layers of operation within each node, from generating and receiving physical signals of various types, to selecting a link for transferring those signals, to the format of information indicated by those signals, to identifying which software application executing on a computer system sends or receives the information. The conceptually different layers of protocols for exchanging information over a network are described in the Open Systems Interconnection (OSI) Reference Model.

Communications between the network nodes are typically effected by exchanging discrete packets of data. Each packet typically comprises (1) header information associated with a particular protocol, and (2) payload information that follows the header information and contains information that may be processed independently of that particular protocol. In some protocols, the packet includes (3) trailer information following the payload and indicating the end of the payload information. The header includes information such as the source of the packet, its destination, the length of the payload, and other properties used by the protocol. Often, the data in the payload for the particular protocol includes a header and payload for a different protocol associated with a different, higher layer of the OSI Reference Model. The header for a particular protocol typically indicates a type for the next protocol contained in its payload. The higher layer protocol is said to be encapsulated in the lower layer protocol. The headers included in a packet traversing multiple heterogeneous networks, such as the Internet, typically include a physical (layer 1) header, a data-link (layer 2) header, an internetwork (layer 3) header and a transport (layer 4) header, and various application (layer 5, layer 6 and layer 7) headers as defined by the OSI Reference Model.

FIG. 2 is a diagram of the components of the ordering platform **103**, according to one embodiment. By way of example, the ordering platform **103** includes one or more components for providing an ordering metric for a multi-dimensional contextual query. It is contemplated that the functions of these components may be combined in one or

more components or performed by other components of equivalent functionality. For example, the functions of these components may be embodied in one or more applications 111 executed on a UE 101. Alternatively, the functions of these components can be embodied in one or more modules of the UE 101, or one or more services 107 on the service platform 109.

In this embodiment, the ordering platform 103 includes a persona module 201, a context module 203, a query module 205, an ordering metric module 207, an analyzer module 209, and a user interface module 211.

The persona module 201 determines one or more personas associated with a user of a UE 101 and/or a UE 101. The persona module 201 determines the one or more personas by, for example, a user or users of the UE 101 entering the information regarding the personas.

The personas represent the possible different combinations of users that can be associated with a specific UE 101. By way of example, a persona can represent the members of a family that are associated with the UE 101. The persona can also represent the members of a group of friends that are associated with the UE 101. The personas allow for the creation of situations associated with the UE 101 that correlate to user contextual attributes that form the multi-dimensional queries, in addition to, for example, context information, to generate a list of results from one or more databases 115 that can later be ordered according to an ordering metric.

In one embodiment, when a UE 101 first interfaces with the ordering platform 103, the persona module 201 determines a complete persona associated with the UE 101. By way of example, the complete persona includes representations of the type (e.g., gender, age group, disability, relationships, etc.) and quantity of the users that could possibly be associated with the UE 101. Such examples include one male parent, one female parent, and two infant daughters; one female parent, one teenage boy, and one infant boy; one disabled friend and a non-disabled friend, etc. The complete persona allows the ordering platform 103 to determine all possible personas that could be associated with the UE 101 based on the possible users associated with the UE 101. Thus, the complete persona allows the ordering platform 103 to determine all of the possible user contextual attributes that could be associated with the UE 101 based on the possible users associated with the UE 101.

In one embodiment, the persona module 201 determines one or more personas associated with users of a UE 101 at the time of querying and/or ordering of the one or more results of a multi-dimensional query. The persona module 201 determines the specific users that are associated with the UE 101, and, therefore, associated with the query and/or the ordering metric, and creates the multi-dimensional query based, at least in part, on the user contextual attributes associated with the persona. By way of example, certain user contextual attributes are associated with male parents, female parents, non-parent adults, teenagers, young children, infants, friends, disabled individuals, non-disabled individuals, etc. By indicating the number and type (e.g., gender, age group, disability, relationships, etc.) of users associated with the UE 101 at the time of the query and/or ordering of the one or more results, the ordering platform 103 automatically generates persona information and associates the persona information with user contextual attributes. By doing so, the persona module 201 allows the ordering platform 103 to have situational awareness over the users that are associated with the UE 101 and who are querying the databases 115.

The context module 203 determines the context associated with the user of the UE 101 and/or the UE 101 that is used to

create the multi-dimensional query and/or used to create the ordering metric. The context may comprise, for example, the current location of the user, a future location of the user based on one or more mapping applications 111 running on the UE 101, a current or future appointment based on one or more calendar applications 111 running on the UE 101, etc. The context module 203 can determine the context from, for example, one or more applications 111 running on the UE 101, one or more modules of the UE 101, one or more sensors 117 associated with the UE 101, one or more services 107 associated with the UE 101, or any combination thereof. The context module 203 can also determine the context associated with the users and/or the UE 101 based on interaction information at a user interface of the UE 101. For example, the UE 101 may include user interfaces that allow the users of the UE 101 to enter context regarding the users and/or the UE 101.

In one embodiment, the context module 203 continuously, periodically, or a combination thereof, determines the context information of the user and/or the UE 101 before and after submitting the multi-dimensional query for creating and updating the ordering metric, as discussed below.

The query module 205 creates the multi-dimensional query based on, for example, one or more of the personas and the context acquired by the persona module 201 and the context module 203. The multi-dimensional query can also be based on, for example, one or more dimensions that are specific by a user of the UE 101.

In one embodiment, one dimension can include a contextual dimension for comparing one or more user contextual attributes to one or more object contextual attributes of one or more objects stored in the databases 115. By way of example, the contextual dimension can represent a normalized value of 0 to 1 that defines the most ideally matched object contextual attribute compared to a user contextual attribute. For example, if the context is location, a value of one is the most preferred location of the user and a value of zero is the least preferred location of the user. Thus, if the user contextual attribute for location is a specific latitude and longitude, or a specific street address, and an object has an object contextual attribute with the same latitude and longitude, or street address, the user contextual attribute and the object contextual attribute would have a normalized relation of one, being the most ideally matched.

In one embodiment, one dimension can include a contextual relation dimension for evaluating one or more relations among the one or more user contextual attributes and the one or more object contextual attributes. The contextual relation dimension specifies weighting information, priority information, or a combination thereof of the one or more user contextual attributes, the one or more object contextual attributes, or a combination thereof. Such relations include, for example, an equivalence relation (Q), a total ordering relation (T), and a partial ordering relation (P). In the case of the equivalence relation Q, the results that comply with the user contextual attributes, the one or more object contextual attributes, or a combination thereof are ordered without any special ordering. In the case of the total ordering relation T, the results are sorted based on the contextual dimension discussed above according to the rank of the user contextual attributes, the one or more object contextual attributes, or a combination thereof. In the case of the partial ordering relation P, the results are sorted based on a partial ordering of the contextual dimension discussed above based on the rank of the user contextual attributes, the one or more object contextual attributes, or a combination thereof.

In one embodiment, one dimension can include an inclusion dimension for determining whether to include respective

ones of the one or more results of the multi-dimensional query in a list presented to the user on the user's device. By way of example, the inclusion dimension determines whether objects that include object contextual attributes that do not conform to the user contextual attributes are included or excluded in a list of objects presented to the user. The inclusion attribute can be, for example, unlisted excluded (X), where the objects that include object contextual attributes that do not conform to the user contextual attributes are excluded from the list, and unlisted included (N), where the objects that include object contextual attributes that do not conform to the user contextual attributes are included in the list but come last. The inclusion dimension can be used, for example, to exclude objects and or object information that is unsupported by a mobile device or that is too large to be transmitted over a network to the mobile device.

The query module 205 also executes the multi-dimensional query on the one of more databases 115.

The ordering metric module 207 creates the ordering metric that is used to order the results of the multi-dimensional query. The ordering metric module 207 determines the ordering metric based, at least in part, on one or more of the personas, one or more contexts, one or more of the user contextual attributes, one or more of the object contextual attributes, and one or more dimensions of the multi-dimensional query. In one embodiment, upon generating a list of results representing objects with corresponding object contextual attributes from the databases 115 based on the multi-dimensional query, the ordering metric module 207 generates an ordering metric to order the objects according to the object contextual attributes in comparison to the one or more of the personas, the one or more contexts, the one or more of the user contextual attributes, and the one or more dimensions of the multi-dimensional query. By way of example, the ordering metric orders the objects according to the object contextual attributes in relation to the user contextual attributes according to the contextual dimension and an equivalence relation Q for the contextual relation dimension. The ordering metric module 207 also continuously, periodically, or a combination thereof, monitors for changes in the one or more of the personas, the one or more contexts, the one or more of the user contextual attributes, the one or more of the object contextual attributes, and the one or more dimensions of the multi-dimensional query to reorder the objects based on a new ordering metric. Thus, by way of example, rather than creating a new multi-dimensional query and executing the multi-dimensional query at the databases 115 every time one of the personas, contexts, user contextual attributes, and/or object contextual attributes changes, the ordering platform rather merely updates the ordering metric and reorders the list of result objects with the corresponding object contextual attributes.

The analyzer module 209 determines the accuracy, the precision, and the resolution of the object contextual attributes stored in the databases 115 for the objects presented as results in response to the multi-dimensional query. The accuracy of the object contextual attributes represents the degree of closeness of the object contextual attribute to the actual (true) value. The precision of the object contextual attributes represents the degree of reproducibility of the measurement of the accuracy. The resolution of the object contextual attributes represents the degree of change that is detected. By way of example, some object contextual attributes are yes or no, such as whether a point of interest includes restrooms, parking, or are handicapped accessible. Such contextual attributes should have high degrees of accuracy and precision because there is only one objective answer

(e.g., either yes or no). However, some object contextual attributes can be represented by a normalized value between 0 and 1, such as whether the restrooms are clean, whether the parking spaces can fit a full-sized automobile, whether a restaurant is family friendly, etc. Thus, the analyzer module 209 determines the accuracy, the precision, and the resolution values stored in the databases 115 corresponding to the objects and the object contextual attributes. In one embodiment, the databases 115 already contain the normalized and/or yes/no attributes corresponding to the object contextual attributes. Thus, the analyzer module 209 simply loads the information from the databases 115 and presents the information to the user along with the objects. In one embodiment, the databases 115 have multiple, raw indications of the accuracy, the precision, and the resolution of the object contextual attributes and the analyzer module 209 determines a normalized value for the raw indications. The analyzer module 209 can handle various types of values, ranging from yes/no, to normalized values, to non-normalized values, such as text string descriptions of the object contextual attributes.

In the case where the analyzer module 209 determines the value, the analyzer module 209 determines the accuracy, the precision, and the resolution of the object contextual attributes based on such values (e.g., yes/no, 0-1) stored in the databases 115 associated with the object contextual attributes. The analyzer module 209 determines the actual values based on a wide range of values according to a weighted sum, a weighted average, or a combination thereof of the actual values stored in the databases 115. The analyzer module 209 also interfaces with the user interface module 211 (discussed below) to present the determined values of the accuracy, the precision, and the resolution of the object contextual attributes to the user when the objects are presented as results of the multi-dimensional query.

The analyzer module 209 also receives information from the user, the user of the UE 101, one or more sensors associated with the UE 101, or a combination thereof to further update the values of the accuracy, the precision, and the resolution of the object contextual attributes. By way of example, if a restroom of a restaurant currently has a value of 0.8 associated with the cleanliness attribute based on two reviews or ratings and a user currently rates the cleanliness according as a 0.4 using the user interface of the UE 101, the analyzer module 209 updates the value of the cleanliness of the restroom as a value of 0.66 at the database 115.

The user interface module 211 provides the user interfaces for interfacing with the ordering platform 103. In one embodiment, where the ordering platform 103 is a separate element of the system 100, the user interface module 211 interfaces with one or more applications 111 running on the UE 101 or one or more hardware modules of the UE 101 to provide the user interface on the UE 101 to interface with the ordering platform 103. In one embodiment, where the functions of the ordering platform 103 are embodied in one or more applications 111 running on the UE 101 or in one or more hardware modules of the UE 101, the user interface module 211 interfaces with one or more other hardware modules of the UE 101 to provide the user interface for manipulating the ordering platform 103.

FIG. 3 is a flowchart of a process 300 for providing an ordering metric for a multi-dimensional contextual query, according to one embodiment. In one embodiment, the ordering platform 103 performs the process 300 and is implemented in, for instance, a chip set including a processor and a memory as shown in FIG. 6.

In step 301, the ordering platform 103 determines one or more personas associated with a device (e.g., the UE 101). As

11

discussed above, the personas include, for example, the quantity and type of users that are associated with a device. By way of example, one device is associated with a family that includes a husband, a wife, and five children. In one embodiment, the persona can further be defined by the gender and age group of the children. For example, the five children may include one adult (e.g., over the age of eighteen) female, one teenage (e.g., the ages of thirteen to seventeen) male, one male child (e.g., the ages of two to twelve), and two infant (e.g., younger than two) females. In one embodiment, there may be no distinction between parents and children. For example, the above family would be represented by seven individuals, including one adult male, two adult females, one teenage male, one child male, and two infant females.

Based on the personas, the ordering platform 103 associates the device with one or more situations that are associated with one or more user contextual attributes and/or contexts associated with the members that comprise the persona. For example, by associating a persona of a device with an infant, the ordering platform 103 can associate the device with situations involving infants and user contextual attributes related to infants, such as whether a restaurant has an infant changing room or table in the restroom, whether a location is infant friendly, etc. By associating a persona of a device with an infant and a mother, the ordering platform can associate the device situations involving an infant and the mother of the infant, such as whether a restaurant allows public breast feeding, whether a restaurant has private seating for mothers with infants, etc. By automatically associated the users and/or the device with the personas and related user contextual attributes, the ordering platform 103 is able to determine and handle a wider range of context information based on situational awareness.

In one embodiment, step 301 is performed each time the device interacts with the ordering platform 103 for providing an ordering metric for ordering results of a multi-dimensional contextual query. For example, a first time a device interacts with the ordering platform 103, the user of the device sets up a persona, which is later deleted after ending the interactions with the ordering platform 103. The second time the user interacts with the ordering platform 103, the user is requested to provide information for the persona representing the individuals associated with the device for that particular query. In one embodiment, step 301 is performed to create a default persona that is used each time a multi-dimensional query is issued for determining results of a context-aware database and providing an ordering metric for ordering the results. When the user of the device desires to receive results for a persona other than the default persona, the user is able to temporarily change the persona to a different persona.

In step 303, the ordering platform 103 determines one or more contexts associated with the user of the device and/or the device. The contexts can be related to or further define user contextual attributes associated with the persona setup in step 301 or can be independent of the persona. For example, the contexts can include the current location of the device, a future location of the device based on an application 111a running on the device (e.g., a navigation application with an active route guidance), the current time, a future time based on an application 111b running on the device (e.g., a calendar application with an active or future appointment), etc. The contexts are acquired from any source within the system 100, such as from one or more applications 111 running on the device, one or more sensors 117 associated with the device, one or more services 107 on the service platform 109, and/or one or more content providers 113.

12

In step 305, the ordering platform 103 creates a multi-dimensional query based on the persona and contexts generated from steps 301 and 303 above. The multi-dimensional contextual query includes the situational information associated with the persona from step 301 and includes the context information acquired in step 303. The multi-dimensional query can be of N dimensions.

As discussed above, one dimension can include a contextual dimension for comparing one or more user contextual attributes to one or more object contextual attributes. Further, one dimension can include a contextual relation dimension for evaluating one or more relations among the one or more user contextual attributes and the one or more object contextual attributes. The contextual relation dimension specifies weighting information, priority information, or a combination thereof of the one or more user contextual attributes, the one or more object contextual attributes, or a combination thereof. One dimension can include an inclusion dimension for determining whether to include respective ones of the one or more results of the multi-dimensional query in a list presented to the user on the user's device. One dimension can include the semantic task that the user is involved in, such as finding a restaurant to eat at with the entire family.

In step 307, the ordering platform 103 executes the multi-dimensional query on at least one context-sensitive database 115a to generate one or more results corresponding to object within the databases 115 with their corresponding object contextual attributes. In step 309, the ordering platform 103 determines at least one ordering metric for ordering the one or more results at the device. As discussed above, the ordering metric can be based on a current persona associated with the device, current context information associated with the device, one or more user contextual attributes, one or more object contextual attributes, and/or the multi-dimensional query. The ordering metric can also be based on interaction information at a user interface of the device according to a user modifying any of the above parameters that can modify the ordering metric. By way of example, the user can interact with the device to change the persona information associate with the users of the device from including an infant to not including an infant. Thus, the ordering metric is then based on a persona that does not include an infant and is able to order the results of the multi-dimensional query differently according to the new persona.

In step 311, the ordering platform 103 orders the results of the query based on the at least one ordering metric. The results are ordered based on the object contextual attributes for each of the objects according to the at least one ordering metric. As discussed above, the at least one ordering metric is based, at least in part, on the current persona by ordering object contextual attributes of the one or more results based on the quantity and type of users associated with the device. By way of example, where the users associated with the device include an infant, the at least one ordering metric is focused towards points of interests associated with infants. The at least one ordering metric is based, at least in part, on the current context associated with the device by ordering the object contextual attributes of the one or more results based on the current context. By way of example, where objects include object contextual attributes concerning hours of operation (e.g., hours a restaurant is open, hours for a sale at a store, hours a park is open), the at least one ordering metric orders the objects according to the object contextual attributes that most closely match the current time or a future time. The at least one ordering metric can also be based, at least in part, on the dimensions of the multi-dimensional query. As discussed above, the dimensions of the multi-dimensional con-

13

textual query can include a contextual dimension for comparing the one or more of user contextual attributes and the one or more object contextual attributes, a contextual relation dimension for evaluating one or more relations among the one or more user contextual attributes and the one or more object contextual attributes, and an inclusion dimension for determining whether to include respective ones of the one or more results in a result list. The multi-dimensional query can also include a dimension including the semantic task that the user is involved in. By way of example, if the user is looking for a particular type of restaurant, the ordering metric can include information regarding the particular type of restaurant to rank objects associated with the particular type of restaurant higher than other restaurants.

In step 313, the ordering platform 103 presents accuracy, precision, and/or resolution information regarding the object contextual attributes of the one or more object results of the query. In one embodiment, the ordering platform 103 presents the accuracy, the precision, and/or the resolution information regarding any number of the results of the one or more results. In one embodiment, the ordering platform 103 presents the accuracy, the precision, and/or the resolution information for only a selected one or more of the results of the query.

As discussed above, the accuracy, the precision, and the resolution of the object contextual attributes are stored in the databases 115. The accuracy, the precision, and the resolution of the object contextual attributes are based on such binary values as yes/no or normalized continuous range values such as between 0 and 1. The ordering platform 103 presents the determined values of the accuracy, the precision, and the resolution of the object contextual attributes to the user when the objects are presented as results of the multi-dimensional query.

In step 315, the ordering platform 103 verifies the accuracy, the precision, and/or the resolution of the information regarding the object contextual attributes based on information inputted from the user. By way of example, a user visiting a restaurant who visits the restroom may wish to leave a review regarding the cleanliness of the restaurant. The ordering platform 103 allows the user to enter values regarding the cleanliness of the restroom, such as yes or no, or a normalized value from 0 to 1 (or 1 to 10, etc.) representing various degrees of cleanliness. The ordering platform 103 also allows the user of the device to enter text string values such as, "The worst restroom I have seen to date" to allow for more specificity regarding the object contextual attribute of cleanliness of the restroom. The ordering platform 103 also allows the user of the device to specify more detailed reviews of, for example, restrooms by reviewing parts of the restroom, such as the sink, the toilet, the towels, the floor, etc. Upon the user entering a review of one or more of the object contextual attributes, the ordering platform 103 updates the accuracy, the precision, and/or the resolution of the object contextual attribute based on, for example, a weighted sum, a weighted average, or any other method to update the information and add to the information stored within the databases 115. Thus, the process 300 allows users to add to the situational-aware, context-sensitive information stored within the databases.

In one embodiment, after step 315, the process 300 proceeds back to step 309 based on an indication that any one of the persona, the context, the user contextual attributes, the object contextual attributes, and/or the dimensions have changed indicating that a new ordering metric is created. In this situation, the new ordering metric newly orders the results of the multi-dimensional query without having to query the databases 115 again. This saves time and resources

14

because the information is already at the mobile device. Alternatively, after step 315, the process 300 proceeds back to step 301 for the entire process to be run again in the event that the user wants to create a new multi-dimensional query and determine a new list of results.

FIGS. 4A-4E are diagrams of user interfaces utilized in the processes of FIG. 3, according to various embodiments. FIG. 4A illustrates a user interface 401a of a device 400 (e.g., a UE 101) based on an initial setup of an application 111a interfacing with the ordering platform 103. As discussed above, the user interface 401a includes inputs 403 that allow a user to indicate, for example, the gender, type and quantity of individuals associated with the device 400 to setup the persona associated with the device. By way of example, the persona associated with the device 400 includes one adult male parent, one adult female parent, an adult female non-parent, one teenage male, one child male, and two infant females. The user interface 401a is illustrated when the user wants update or create a persona.

FIG. 4B illustrates a user interface 401b of an application 111a running on a device 400 interfacing with the ordering platform 103. The user interface 401b includes one or more indicators 405 that allow a user to begin a query to determine, for examples, points of interest that the user may wish to visit. By way of example, the user may select the "EAT" indicator 405 to find points of interest involving places to eat. Upon selecting one of the indicators 405, the user interface indicates for the ordering platform 103 to execute a multi-dimensional query on the databases 115 to generate a list of results (e.g., objects) that correspond to the persona and context of associated with the user and/or the device.

FIG. 4C illustrates the user interface 401c of an application 111a running on a device 400 interfacing with the ordering platform 103 including user contextual attributes 407 (and correspondingly object contextual attributes) associated with the determined persona associated with the device 400 that aid the user in finding relevant points of interest. By way of example, if the persona of the user of the device includes a mother with her infant child, the user contextual attributes associated with, for example, parking, a baby room and clean points of interest are determined as the most relevant contextual attributes the user is interested based on the persona. Scrolling down the list of contextual attributes 407 allows the user to select or de-select other contextual attributes, such as, for example, points of interest that are not pricey, points of interest that are open late, and points of interest that have a kids' menu for modify the user contextual attributes associated with the user and/or the device to generate and/or modify the ordering metric.

FIG. 4D illustrates the user interface 401d of an application 111a running on a device 400 interfacing with the ordering platform 103 illustrating detailed information regarding a selected point of interest. Indicator 409 illustrates the specific information regarding the point of interest, such as the name (e.g., Walker Pizza), address (221 Major St., Chicago, Ill.), and phone number (e.g., 312-555-8754) of the point of interest. Included within indicator 409 is indicator 411 that allows a user of the device 400 to enter a review for the particular point of interest.

FIG. 4E illustrates the user interface 401e of an application 111a running on a device 400 interfacing with the ordering platform 103 illustrating the interface for leaving a review of a particular point of interest. By way of example, the user interface 401e includes indicator 413 that includes the object contextual attributes associated with the particular point of interest. As illustrated, a user of the device 400 is allowed to enter reviews of the point of interest based on the particular

15

object contextual attributes by which the point of interest was selected for presentation to the user. In one embodiment, the user can also enter reviews of the object contextual attributes that were not used in selecting the particular object, such as the noise level of a restaurant. As discussed above, the particular point of interest Walker Pizza was selected because the point of interest had object contextual attributes of parking, a baby room, and cleanliness based on the checkmarks of the contextual attributes **407** illustrated in FIG. **4C**. Indicators **415**, **417** and **419** illustrate exemplary ratings of the point of interest. By way of example, the user of the device **400** can indicate whether an object contextual attribute actually existed based on the inclusion of Y for “yes” or N for “no.” The user of the device **400** can also give a numerical value of the object contextual attribute, such as 1-10 representing highly agree (e.g., 1) and highly disagree (e.g., 10) and the different values there between. In one embodiment, the user can also leave text strings regarding the reviews of the object contextual attributes of the point of interest. Based on the above, the ordering platform **103** enters the information entered by the user at the databases **115** to modify the accuracy, the precision and the resolution of the particular object contextual attribute, thus enriching the datasets at the databases **115**.

The processes described herein for providing an ordering metric of a multi-dimensional contextual query may be advantageously implemented via software, hardware, firmware or a combination of software and/or firmware and/or hardware. For example, the processes described herein, may be advantageously implemented via processor(s), Digital Signal Processing (DSP) chip, an Application Specific Integrated Circuit (ASIC), Field Programmable Gate Arrays (FPGAs), etc. Such exemplary hardware for performing the described functions is detailed below.

FIG. **5** illustrates a computer system **500** upon which an embodiment of the invention may be implemented. Although computer system **500** is depicted with respect to a particular device or equipment, it is contemplated that other devices or equipment (e.g., network elements, servers, etc.) within FIG. **5** can deploy the illustrated hardware and components of system **500**. Computer system **500** is programmed (e.g., via computer program code or instructions) to provide ordering metric of a multi-dimensional contextual query as described herein and includes a communication mechanism such as a bus **510** for passing information between other internal and external components of the computer system **500**. Information (also called data) is represented as a physical expression of a measurable phenomenon, typically electric voltages, but including, in other embodiments, such phenomena as magnetic, electromagnetic, pressure, chemical, biological, molecular, atomic, sub-atomic and quantum interactions. For example, north and south magnetic fields, or a zero and non-zero electric voltage, represent two states (0, 1) of a binary digit (bit). Other phenomena can represent digits of a higher base. A superposition of multiple simultaneous quantum states before measurement represents a quantum bit (qubit). A sequence of one or more digits constitutes digital data that is used to represent a number or code for a character. In some embodiments, information called analog data is represented by a near continuum of measurable values within a particular range. Computer system **500**, or a portion thereof, constitutes a means for performing one or more steps of providing an ordering metric of a multi-dimensional contextual query.

A bus **510** includes one or more parallel conductors of information so that information is transferred quickly among devices coupled to the bus **510**. One or more processors **502** for processing information are coupled with the bus **510**.

16

A processor (or multiple processors) **502** performs a set of operations on information as specified by computer program code related to provide an ordering metric of a multi-dimensional contextual query. The computer program code is a set of instructions or statements providing instructions for the operation of the processor and/or the computer system to perform specified functions. The code, for example, may be written in a computer programming language that is compiled into a native instruction set of the processor. The code may also be written directly using the native instruction set (e.g., machine language). The set of operations include bringing information in from the bus **510** and placing information on the bus **510**. The set of operations also typically include comparing two or more units of information, shifting positions of units of information, and combining two or more units of information, such as by addition or multiplication or logical operations like OR, exclusive OR (XOR), and AND. Each operation of the set of operations that can be performed by the processor is represented to the processor by information called instructions, such as an operation code of one or more digits. A sequence of operations to be executed by the processor **502**, such as a sequence of operation codes, constitute processor instructions, also called computer system instructions or, simply, computer instructions. Processors may be implemented as mechanical, electrical, magnetic, optical, chemical or quantum components, among others, alone or in combination.

Computer system **500** also includes a memory **504** coupled to bus **510**. The memory **504**, such as a random access memory (RAM) or any other dynamic storage device, stores information including processor instructions for providing an ordering metric of a multi-dimensional contextual query. Dynamic memory allows information stored therein to be changed by the computer system **500**. RAM allows a unit of information stored at a location called a memory address to be stored and retrieved independently of information at neighboring addresses. The memory **504** is also used by the processor **502** to store temporary values during execution of processor instructions. The computer system **500** also includes a read only memory (ROM) **506** or any other static storage device coupled to the bus **510** for storing static information, including instructions, that is not changed by the computer system **500**. Some memory is composed of volatile storage that loses the information stored thereon when power is lost. Also coupled to bus **510** is a non-volatile (persistent) storage device **508**, such as a magnetic disk, optical disk or flash card, for storing information, including instructions, that persists even when the computer system **500** is turned off or otherwise loses power.

Information, including instructions for providing an ordering metric of a multi-dimensional contextual query, is provided to the bus **510** for use by the processor from an external input device **512**, such as a keyboard containing alphanumeric keys operated by a human user, a microphone, an Infrared (IR) remote control, a joystick, a game pad, a stylus pen, a touch screen, or a sensor. A sensor detects conditions in its vicinity and transforms those detections into physical expression compatible with the measurable phenomenon used to represent information in computer system **500**. Other external devices coupled to bus **510**, used primarily for interacting with humans, include a display device **514**, such as a cathode ray tube (CRT), a liquid crystal display (LCD), a light emitting diode (LED) display, an organic LED (OLED) display, a plasma screen, or a printer for presenting text or images, and a pointing device **516**, such as a mouse, a trackball, cursor direction keys, or a motion sensor, for controlling a position of a small cursor image presented on the display

514 and issuing commands associated with graphical elements presented on the display **514**. In some embodiments, for example, in embodiments in which the computer system **500** performs all functions automatically without human input, one or more of external input device **512**, display device **514** and pointing device **516** is omitted.

In the illustrated embodiment, special purpose hardware, such as an application specific integrated circuit (ASIC) **520**, is coupled to bus **510**. The special purpose hardware is configured to perform operations not performed by processor **502** quickly enough for special purposes. Examples of ASICs include graphics accelerator cards for generating images for display **514**, cryptographic boards for encrypting and decrypting messages sent over a network, speech recognition, and interfaces to special external devices, such as robotic arms and medical scanning equipment that repeatedly perform some complex sequence of operations that are more efficiently implemented in hardware.

Computer system **500** also includes one or more instances of a communications interface **570** coupled to bus **510**. Communication interface **570** provides a one-way or two-way communication coupling to a variety of external devices that operate with their own processors, such as printers, scanners and external disks. In general the coupling is with a network link **578** that is connected to a local network **580** to which a variety of external devices with their own processors are connected. For example, communication interface **570** may be a parallel port or a serial port or a universal serial bus (USB) port on a personal computer. In some embodiments, communications interface **570** is an integrated services digital network (ISDN) card or a digital subscriber line (DSL) card or a telephone modem that provides an information communication connection to a corresponding type of telephone line. In some embodiments, a communication interface **570** is a cable modem that converts signals on bus **510** into signals for a communication connection over a coaxial cable or into optical signals for a communication connection over a fiber optic cable. As another example, communications interface **570** may be a local area network (LAN) card to provide a data communication connection to a compatible LAN, such as Ethernet. Wireless links may also be implemented. For wireless links, the communications interface **570** sends or receives or both sends and receives electrical, acoustic or electromagnetic signals, including infrared and optical signals, that carry information streams, such as digital data. For example, in wireless handheld devices, such as mobile telephones like cell phones, the communications interface **570** includes a radio band electromagnetic transmitter and receiver called a radio transceiver. In certain embodiments, the communications interface **570** enables connection to the communication network **105** for providing an ordering metric of a multi-dimensional contextual query to the UE **101**.

The term “computer-readable medium” as used herein refers to any medium that participates in providing information to processor **502**, including instructions for execution. Such a medium may take many forms, including, but not limited to computer-readable storage medium (e.g., non-volatile media, volatile media), and transmission media. Non-transitory media, such as non-volatile media, include, for example, optical or magnetic disks, such as storage device **508**. Volatile media include, for example, dynamic memory **504**. Transmission media include, for example, twisted pair cables, coaxial cables, copper wire, fiber optic cables, and carrier waves that travel through space without wires or cables, such as acoustic waves and electromagnetic waves, including radio, optical and infrared waves. Signals include man-made transient variations in amplitude, frequency,

phase, polarization or other physical properties transmitted through the transmission media. Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, any other magnetic medium, a CD-ROM, CDRW, DVD, any other optical medium, punch cards, paper tape, optical mark sheets, any other physical medium with patterns of holes or other optically recognizable indicia, a RAM, a PROM, an EPROM, a FLASH-EPROM, an EEPROM, a flash memory, any other memory chip or cartridge, a carrier wave, or any other medium from which a computer can read. The term computer-readable storage medium is used herein to refer to any computer-readable medium except transmission media.

Logic encoded in one or more tangible media includes one or both of processor instructions on a computer-readable storage media and special purpose hardware, such as ASIC **520**.

Network link **578** typically provides information communication using transmission media through one or more networks to other devices that use or process the information. For example, network link **578** may provide a connection through local network **580** to a host computer **582** or to equipment **584** operated by an Internet Service Provider (ISP). ISP equipment **584** in turn provides data communication services through the public, world-wide packet-switching communication network of networks now commonly referred to as the Internet **590**.

A computer called a server host **592** connected to the Internet hosts a process that provides a service in response to information received over the Internet. For example, server host **592** hosts a process that provides information representing video data for presentation at display **514**. It is contemplated that the components of system **500** can be deployed in various configurations within other computer systems, e.g., host **582** and server **592**.

At least some embodiments of the invention are related to the use of computer system **500** for implementing some or all of the techniques described herein. According to one embodiment of the invention, those techniques are performed by computer system **500** in response to processor **502** executing one or more sequences of one or more processor instructions contained in memory **504**. Such instructions, also called computer instructions, software and program code, may be read into memory **504** from another computer-readable medium such as storage device **508** or network link **578**. Execution of the sequences of instructions contained in memory **504** causes processor **502** to perform one or more of the method steps described herein. In alternative embodiments, hardware, such as ASIC **520**, may be used in place of or in combination with software to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware and software, unless otherwise explicitly stated herein.

The signals transmitted over network link **578** and other networks through communications interface **570**, carry information to and from computer system **500**. Computer system **500** can send and receive information, including program code, through the networks **580**, **590** among others, through network link **578** and communications interface **570**. In an example using the Internet **590**, a server host **592** transmits program code for a particular application, requested by a message sent from computer **500**, through Internet **590**, ISP equipment **584**, local network **580** and communications interface **570**. The received code may be executed by processor **502** as it is received, or may be stored in memory **504** or in storage device **508** or any other non-volatile storage for later

execution, or both. In this manner, computer system **500** may obtain application program code in the form of signals on a carrier wave.

Various forms of computer readable media may be involved in carrying one or more sequence of instructions or data or both to processor **502** for execution. For example, instructions and data may initially be carried on a magnetic disk of a remote computer such as host **582**. The remote computer loads the instructions and data into its dynamic memory and sends the instructions and data over a telephone line using a modem. A modem local to the computer system **500** receives the instructions and data on a telephone line and uses an infra-red transmitter to convert the instructions and data to a signal on an infra-red carrier wave serving as the network link **578**. An infrared detector serving as communications interface **570** receives the instructions and data carried in the infrared signal and places information representing the instructions and data onto bus **510**. Bus **510** carries the information to memory **504** from which processor **502** retrieves and executes the instructions using some of the data sent with the instructions. The instructions and data received in memory **504** may optionally be stored on storage device **508**, either before or after execution by the processor **502**.

FIG. **6** illustrates a chip set or chip **600** upon which an embodiment of the invention may be implemented. Chip set **600** is programmed to provide an ordering metric of a multi-dimensional contextual query as described herein and includes, for instance, the processor and memory components described with respect to FIG. **5** incorporated in one or more physical packages (e.g., chips). By way of example, a physical package includes an arrangement of one or more materials, components, and/or wires on a structural assembly (e.g., a baseboard) to provide one or more characteristics such as physical strength, conservation of size, and/or limitation of electrical interaction. It is contemplated that in certain embodiments the chip set **600** can be implemented in a single chip. It is further contemplated that in certain embodiments the chip set or chip **600** can be implemented as a single "system on a chip." It is further contemplated that in certain embodiments a separate ASIC would not be used, for example, and that all relevant functions as disclosed herein would be performed by a processor or processors. Chip set or chip **600**, or a portion thereof, constitutes a means for performing one or more steps of providing user interface navigation information associated with the availability of functions. Chip set or chip **600**, or a portion thereof, constitutes a means for performing one or more steps of providing an ordering metric of a multi-dimensional contextual query.

In one embodiment, the chip set or chip **600** includes a communication mechanism such as a bus **601** for passing information among the components of the chip set **600**. A processor **603** has connectivity to the bus **601** to execute instructions and process information stored in, for example, a memory **605**. The processor **603** may include one or more processing cores with each core configured to perform independently. A multi-core processor enables multiprocessing within a single physical package. Examples of a multi-core processor include two, four, eight, or greater numbers of processing cores. Alternatively or in addition, the processor **603** may include one or more microprocessors configured in tandem via the bus **601** to enable independent execution of instructions, pipelining, and multithreading. The processor **603** may also be accompanied with one or more specialized components to perform certain processing functions and tasks such as one or more digital signal processors (DSP) **607**, or one or more application-specific integrated circuits (ASIC) **609**. A DSP **607** typically is configured to process real-world

signals (e.g., sound) in real time independently of the processor **603**. Similarly, an ASIC **609** can be configured to perform specialized functions not easily performed by a more general purpose processor. Other specialized components to aid in performing the inventive functions described herein may include one or more field programmable gate arrays (FPGA), one or more controllers, or one or more other special-purpose computer chips.

In one embodiment, the chip set or chip **600** includes merely one or more processors and some software and/or firmware supporting and/or relating to and/or for the one or more processors.

The processor **603** and accompanying components have connectivity to the memory **605** via the bus **601**. The memory **605** includes both dynamic memory (e.g., RAM, magnetic disk, writable optical disk, etc.) and static memory (e.g., ROM, CD-ROM, etc.) for storing executable instructions that when executed perform the inventive steps described herein to provide an ordering metric of a multi-dimensional contextual query. The memory **605** also stores the data associated with or generated by the execution of the inventive steps.

FIG. **7** is a diagram of exemplary components of a mobile terminal (e.g., handset) for communications, which is capable of operating in the system of FIG. **1**, according to one embodiment. In some embodiments, mobile terminal **701**, or a portion thereof, constitutes a means for performing one or more steps of providing an ordering metric of a multi-dimensional contextual query. Generally, a radio receiver is often defined in terms of front-end and back-end characteristics. The front-end of the receiver encompasses all of the Radio Frequency (RF) circuitry whereas the back-end encompasses all of the base-band processing circuitry. As used in this application, the term "circuitry" refers to both: (1) hardware-only implementations (such as implementations in only analog and/or digital circuitry), and (2) to combinations of circuitry and software (and/or firmware) (such as, if applicable to the particular context, to a combination of processor(s), including digital signal processor(s), software, and memory(ies) that work together to cause an apparatus, such as a mobile phone or server, to perform various functions). This definition of "circuitry" applies to all uses of this term in this application, including in any claims. As a further example, as used in this application and if applicable to the particular context, the term "circuitry" would also cover an implementation of merely a processor (or multiple processors) and its (or their) accompanying software/or firmware. The term "circuitry" would also cover if applicable to the particular context, for example, a baseband integrated circuit or applications processor integrated circuit in a mobile phone or a similar integrated circuit in a cellular network device or other network devices.

Pertinent internal components of the telephone include a Main Control Unit (MCU) **703**, a Digital Signal Processor (DSP) **705**, and a receiver/transmitter unit including a microphone gain control unit and a speaker gain control unit. A main display unit **707** provides a display to the user in support of various applications and mobile terminal functions that perform or support the steps of providing an ordering metric of a multi-dimensional contextual query. The display **707** includes display circuitry configured to display at least a portion of a user interface of the mobile terminal (e.g., mobile telephone). Additionally, the display **707** and display circuitry are configured to facilitate user control of at least some functions of the mobile terminal. An audio function circuitry **709** includes a microphone **711** and microphone amplifier that amplifies the speech signal output from the microphone

711. The amplified speech signal output from the microphone 711 is fed to a coder/decoder (CODEC) 713.

A radio section 715 amplifies power and converts frequency in order to communicate with a base station, which is included in a mobile communication system, via antenna 717. The power amplifier (PA) 719 and the transmitter/modulation circuitry are operationally responsive to the MCU 703, with an output from the PA 719 coupled to the duplexer 721 or circulator or antenna switch, as known in the art. The PA 719 also couples to a battery interface and power control unit 720.

In use, a user of mobile terminal 701 speaks into the microphone 711 and his or her voice along with any detected background noise is converted into an analog voltage. The analog voltage is then converted into a digital signal through the Analog to Digital Converter (ADC) 723. The control unit 703 routes the digital signal into the DSP 705 for processing therein, such as speech encoding, channel encoding, encrypting, and interleaving. In one embodiment, the processed voice signals are encoded, by units not separately shown, using a cellular transmission protocol such as enhanced data rates for global evolution (EDGE), general packet radio service (GPRS), global system for mobile communications (GSM), Internet protocol multimedia subsystem (IMS), universal mobile telecommunications system (UMTS), etc., as well as any other suitable wireless medium, e.g., microwave access (WiMAX), Long Term Evolution (LTE) networks, code division multiple access (CDMA), wideband code division multiple access (WCDMA), wireless fidelity (WiFi), satellite, and the like, or any combination thereof.

The encoded signals are then routed to an equalizer 725 for compensation of any frequency-dependent impairments that occur during transmission through the air such as phase and amplitude distortion. After equalizing the bit stream, the modulator 727 combines the signal with a RF signal generated in the RF interface 729. The modulator 727 generates a sine wave by way of frequency or phase modulation. In order to prepare the signal for transmission, an up-converter 731 combines the sine wave output from the modulator 727 with another sine wave generated by a synthesizer 733 to achieve the desired frequency of transmission. The signal is then sent through a PA 719 to increase the signal to an appropriate power level. In practical systems, the PA 719 acts as a variable gain amplifier whose gain is controlled by the DSP 705 from information received from a network base station. The signal is then filtered within the duplexer 721 and optionally sent to an antenna coupler 735 to match impedances to provide maximum power transfer. Finally, the signal is transmitted via antenna 717 to a local base station. An automatic gain control (AGC) can be supplied to control the gain of the final stages of the receiver. The signals may be forwarded from there to a remote telephone which may be another cellular telephone, any other mobile phone or a land-line connected to a Public Switched Telephone Network (PSTN), or other telephony networks.

Voice signals transmitted to the mobile terminal 701 are received via antenna 717 and immediately amplified by a low noise amplifier (LNA) 737. A down-converter 739 lowers the carrier frequency while the demodulator 741 strips away the RF leaving only a digital bit stream. The signal then goes through the equalizer 725 and is processed by the DSP 705. A Digital to Analog Converter (DAC) 743 converts the signal and the resulting output is transmitted to the user through the speaker 745, all under control of a Main Control Unit (MCU) 703 which can be implemented as a Central Processing Unit (CPU).

The MCU 703 receives various signals including input signals from the keyboard 747. The keyboard 747 and/or the

MCU 703 in combination with other user input components (e.g., the microphone 711) comprise a user interface circuitry for managing user input. The MCU 703 runs a user interface software to facilitate user control of at least some functions of the mobile terminal 701 to provide an ordering metric of a multi-dimensional contextual query. The MCU 703 also delivers a display command and a switch command to the display 707 and to the speech output switching controller, respectively. Further, the MCU 703 exchanges information with the DSP 705 and can access an optionally incorporated SIM card 749 and a memory 751. In addition, the MCU 703 executes various control functions required of the terminal. The DSP 705 may, depending upon the implementation, perform any of a variety of conventional digital processing functions on the voice signals. Additionally, DSP 705 determines the background noise level of the local environment from the signals detected by microphone 711 and sets the gain of microphone 711 to a level selected to compensate for the natural tendency of the user of the mobile terminal 701.

The CODEC 713 includes the ADC 723 and DAC 743. The memory 751 stores various data including call incoming tone data and is capable of storing other data including music data received via, e.g., the global Internet. The software module could reside in RAM memory, flash memory, registers, or any other form of writable storage medium known in the art. The memory device 751 may be, but not limited to, a single memory, CD, DVD, ROM, RAM, EEPROM, optical storage, magnetic disk storage, flash memory storage, or any other non-volatile storage medium capable of storing digital data.

An optionally incorporated SIM card 749 carries, for instance, important information, such as the cellular phone number, the carrier supplying service, subscription details, and security information. The SIM card 749 serves primarily to identify the mobile terminal 701 on a radio network. The card 749 also contains a memory for storing a personal telephone number registry, text messages, and user specific mobile terminal settings.

While the invention has been described in connection with a number of embodiments and implementations, the invention is not so limited but covers various obvious modifications and equivalent arrangements, which fall within the purview of the appended claims. Although features of the invention are expressed in certain combinations among the claims, it is contemplated that these features can be arranged in any combination and order.

What is claimed is:

1. A method comprising facilitating a processing of and/or processing (1) data and/or (2) information and/or (3) at least one signal, the (1) data and/or (2) information and/or (3) at least one signal based, at least in part, on the following:

a determination of a multi-dimensional query associated with at least one user device, wherein the multi-dimensional query specifies, at least in part, one or more personas, based, at least in part, on more than one person, associated with the at least one user device;

an execution of the multi-dimensional query on at least one context-sensitive database to generate one or more results; and

a determination of at least one ordering metric for the one or more results based, at least in part, on one or more user contextual attributes of the at least one user device.

2. A method of claim 1, wherein the (1) data and/or (2) information and/or (3) at least one signal are further based, at least in part, on the following:

a determination of one or more object contextual attributes associated with the one or more results,

23

wherein the at least one ordering metric is further based, at least in part, on the one or more object contextual attributes.

3. A method of claim 2, wherein the (1) data and/or (2) information and/or (3) at least one signal are further based, at least in part, on the following:

a determination of an accuracy, a precision, a resolution, or a combination thereof of the one or more object contextual attributes; and

a display of one or more representations of the accuracy, the precision, the resolution, or a combination thereof.

4. A method of claim 3, wherein the (1) data and/or (2) information and/or (3) at least one signal are further based, at least in part, on the following:

a determination of the accuracy, the precision, the resolution, or a combination thereof based, at least in part, on sensor information, survey information, user update information, or a combination thereof.

5. A method of claim 4, wherein the (1) data and/or (2) information and/or (3) at least one signal are further based, at least in part, on the following:

an updating of the one or more object contextual attributes at the at least one context-sensitive database based on the determined accuracy, the determined precision, the determined resolution, or a combination thereof.

6. A method of claim 2, wherein the at least one ordering metric is based, at least in part, on a weighted sum, a weighted average, or a combination thereof of one or more dimensions of the multi-dimensional query.

7. A method of claim 6, wherein the one or more dimensions include, at least in part, (i) a contextual dimension for comparing the one or more user contextual attributes and the one or more object contextual attributes, (ii) a contextual relation dimension for evaluating one or more relations among the one or more user contextual attributes and the one or more object contextual attributes, and (iii) an inclusion dimension for determining whether to include respective ones of the one or more results in a result list.

8. A method of claim 7, wherein the contextual relation dimension specifies weighting information, priority information, or a combination thereof for the one or more user contextual attributes, the one or more object contextual attributes, or combination thereof.

9. A method of claim 1, wherein the multi-dimensional query specifies, at least in part, one or more contexts associated with the at least one user device, and the (1) data and/or (2) information and/or (3) at least one signal are further based, at least in part, on the following:

a processing of the one or more user contextual attributes to select from among the one or more personas, the one or more contexts, or a combination thereof,

wherein the at least one ordering metric is based, at least in part, on the selected one or more personas, the selected one or more contexts, or a combination thereof.

10. A method of claim 1, wherein the (1) data and/or (2) information and/or (3) at least one signal are further based, at least in part, on the following:

a determination of contextual information, interaction information at a user interface of the at least one user device, or a combination thereof associated with the at least one user device; and

a processing of the contextual information, the interaction information, or a combination thereof to determine, at least in part, the one or more user contextual attributes.

24

11. An apparatus comprising:

at least one processor; and

at least one memory including computer program code for one or more programs,

the at least one memory and the computer program code configured to, with the at least one processor, cause the apparatus to perform at least the following,

determine a multi-dimensional query associated with at least one user device, wherein the multi-dimensional query specifies, at least in part, one or more personas, based, at least in part, on more than one person, associated with the at least one user device;

cause, at least in part, an execution of the multi-dimensional query on at least one context-sensitive database to generate one or more results; and

determine at least one ordering metric for the one or more results based, at least in part, on one or more user contextual attributes of the at least one user device.

12. An apparatus of claim 11, wherein the apparatus is further caused to:

determine one or more object contextual attributes associated with the one or more results,

wherein the at least one ordering metric is further based, at least in part, on the one or more object contextual attributes.

13. An apparatus of claim 12, wherein the apparatus is further caused to:

determine an accuracy, a precision, a resolution, or a combination thereof of the one or more object contextual attributes; and

cause, at least in part, a display of one or more representations of the accuracy, the precision, the resolution, or a combination thereof.

14. An apparatus of claim 13, wherein the apparatus is further caused to:

determine the accuracy, the precision, the resolution, or a combination thereof based, at least in part, on sensor information, survey information, user update information, or a combination thereof.

15. An apparatus of claim 14, wherein the apparatus is further caused to:

cause, at least in part, an updating of the one or more object contextual attributes at the at least one context-sensitive database based on the determined accuracy, the determined precision, the determined resolution, or a combination thereof.

16. An apparatus of claim 12, wherein the at least one ordering metric is based, at least in part, on a weighted sum, a weighted average, or a combination thereof of one or more dimensions of the multi-dimensional query.

17. An apparatus of claim 16, wherein the one or more dimensions include, at least in part, (i) a contextual dimension for comparing the one or more user contextual attributes and the one or more object contextual attributes, (ii) a contextual relation dimension for evaluating one or more relations among the one or more user contextual attributes and the one or more object contextual attributes, and (iii) an inclusion dimension for determining whether to include respective ones of the one or more results in a result list.

18. An apparatus of claim 17, wherein the contextual relation dimension specifies weighting information, priority information, or a combination thereof for the one or more user contextual attributes, the one or more object contextual attributes, or combination thereof.

19. An apparatus of claim 11, wherein the multi-dimensional query specifies, at least in part, one or more contexts associated with the at least one user device, and the apparatus is further caused to:

process and/or facilitate a processing of the one or more user contextual attributes to select from among the one or more personas, the one or more contexts, or a combination thereof,

wherein the at least one ordering metric is based, at least in part, on the selected one or more personas, the selected one or more contexts, or a combination thereof.

20. An apparatus of claim 11, wherein the apparatus is further caused to:

determine contextual information, interaction information at a user interface of the at least one user device, or a combination thereof associated with the at least one user device; and

process and/or facilitate a processing of the contextual information, the interaction information, or a combination thereof to determine, at least in part, the one or more user contextual attributes.

* * * * *